

IMPROVING THE PRODUCTIVITY OF
SYSTEMS AND SOFTWARE IMPLEMENTATION

INPUT
LIBRARY

ABOUT INPUT

THE COMPANY

INPUT provides planning information, analysis, and recommendations to managers and executives in the information processing industries. Through market research, technology forecasting, and competitive analysis, INPUT supports client management in making informed decisions. Continuing services are provided to users and vendors of computers, computer products and services.

The company carries out depth research. Clients on important members analyze data, then develop innovative ideas. Clients receive access to data on a continuous basis.

Many of INPUT's offices have nearly 20 years of special experience in senior management consulting, marketing, or product development. This enables INPUT to handle complex business problems.

Formed in 1974, INPUT is a leading international firm. Clients include the largest and most successful companies.

UNITED STATES, West Coast

2471 East Bayshore Road
Suite 600
Palo Alto, California 94303
(415) 493-1600
Telex 171407

4676 Admiralty Way
#401 C
Marina Del Rey, California 90291
1230

X-PRO

Heidenrich, George
Tyler, Tim.

Improving the Productivity of
Systems and Software Implementation

UNITED STATES, East Coast

West-1
New Jersey 07662

DM

4th Floor)

ia
, 7-9 Merriwa Street
Gordon N.S.W. 2072

L

JAPAN

INPUT Japan
Suite 1106
7-7-26 Nishi-Shinjuku
Tokyo
Japan 160
(03) 371-3082

IMPROVING THE PRODUCTIVITY OF
SYSTEMS AND SOFTWARE IMPLEMENTATION

A MULTICLIENT STUDY

NOVEMBER 1980

INPUT LIBRARY



Digitized by the Internet Archive
in 2015

https://archive.org/details/improvingproductunse_3

IMPROVING THE PRODUCTIVITY OF SYSTEMS AND SOFTWARE IMPLEMENTATION

TABLE OF CONTENTS

	<u>Page</u>
I INTRODUCTION	I
II EXECUTIVE SUMMARY	5
A. Current "Solutions" Do Not Solve Productivity Problems	5
B. Current Perceptions Of The Problem Are Too Limited	9
C. A Productivity Strategy Is Required	12
D. There Is No Single Strategy For All Organizations	15
E. Stages Of Productivity Development: A Key Concept	15
F. Self-Assessment	16
III DIMENSIONS OF THE PRODUCTIVITY PROBLEM	19
A. Perception And Definition	19
1. Where Does The Problem Lie?	19
2. Historical Perspective	21
3. Current "Solutions"	24
B. Productivity Issues	26
1. Symptoms Of The Problem	26
2. Inhibiting Factors	26
3. Goals And Mechanisms	30
4. How The EDP Department Believes Goals Should Be Achieved	32
5. Current Actions To Improve Productivity	37
6. Summary	41
7. A Preliminary Diagnosis And Definition	42
C. Productivity Through EDP User Effectiveness	44
1. The Forgotten User	44
2. The Largest Software Productivity Failure	45
3. The Critical Needs Definition Process	48
D. Productivity Through Effective Management	52
1. Management Interrelationships	52
2. EDP Management's External Role	54
3. EDP Management's Internal Role	55
4. Management Balance	58
E. Productivity Improvements Through EDP Personnel Effectiveness	58
1. Personnel Utilization - A Key Factor	58
2. Personnel Shortage	64

	<u>Page</u>
3. Strategies For Dealing With Personnel Shortages And Recruitment Problems	71
4. Turnover	74
5. Morale And Motivation	81
F. A Productivity Strategy	83
G. Management By Objectives: A First Step	92
IV THE PRODUCTIVITY SELF-ANALYSIS MATRIX	95
A. Stages Of Productivity Development	95
B. The Characteristics Of Each Stage Of Software Productivity Development	108
C. Identifying Productivity Determinants, Instruments And Results	117
1. Determinants Of Productivity	118
2. Instruments Of Productivity	119
3. Results Of Productivity	123
D. Self-Analysis Matrix Diagnosis	123
E. Self-Analysis Matrix Prescription	126
F. Activity-Oriented Audit Of Productivity	130
V PROFILE OF PROCESS IMPROVEMENT AIDS	135
A. Rationale For Categorization	135
1. General	135
2. The Three-Dimensional "Problem Space"	135
3. "Classical" Versus "Emerging"	139
4. Applicability Relative To Life Cycle Phases	142
5. Impact Relative To Life Cycle Phases	144
B. Life Cycle Cost Improvements Versus Tool Costs	147
1. Structured Tableau	152
2. PDL	154
3. SADT	156
4. PSL/PSA	157
5. Reusable Code	157
6. PWB/UNIX	158
7. PRIDE/Logik	158
C. Feature Analysis Of Selected Tools And Techniques	159
1. On-Line Program Development Aids	159
2. Programmer's Workbench	162
3. Structured Programming/Design/Analysis	167
4. Data-Driven Structured Methodologies	175
5. SADT (Structured Analysis Design Tool)	177
6. HIPO (Hierarchy Plus Input-Process-Output)	182
7. Structured Tableau	185
8. System Development Methodologies (SDMs)	185
9. PRIDE/Logik-ADF	190
10. Requirements Languages	193
11. PDL	200

	<u>Page</u>
12. Reusable Code	201
13. Menu-Driven Programming	206
D. Future Directions	209
VI SELECTED CASE HISTORIES	219
A. Case History #1 - Successful Stage Two Manufacturer	219
1. The Company	219
2. EDP Organization	220
3. Application Types	221
4. EDP Management Description	221
5. Control Mechanism	222
6. Performance Characteristics	222
7. Quality Control	223
8. User Involvement	223
9. Programming Resources	224
10. Productivity Tools	224
11. Investment Payoff	225
B. Case History #2 - Successful Stage One Manufacturer	226
1. The Company	226
2. EDP Organization	226
3. Application Types	227
4. EDP Management Description	227
5. Control Mechanism	228
6. Performance Characteristics	229
7. Quality Control	229
8. User Involvement	230
9. Programming Resources	230
10. Productivity Tools	231
11. Investment Payoff	231
C. Case History #3 - Successful Stage Three Manufacturer	232
1. The Company	232
2. EDP Organization	233
3. Application Type	233
4. EDP Management Description	234
5. Control Mechanism	234
6. Performance Characteristics	235
7. Quality Control	236
8. User Involvement	236
9. Programming Resources	237
10. Productivity Tools	237
11. Investment Payoff	240
D. Case History #4 - Unsuccessful Stage Zero Transportation Company	241
1. The Company	241
2. EDP Organization	241
3. Application Type	242
4. EDP Management Description	242
5. Control Mechanism	243

	<u>Page</u>
6. Performance Characteristics	244
7. Quality Control	244
8. User Involvement	245
9. Programming Resources	245
10. Productivity Tools	246
11. Investment Payoff	246
E. Cast History #5: Optimizer III, Mark IV	247
1. The Company	247
2. EDP Environment	247
3. Productivity Problems/Selected Techniques	247
4. Technique Evaluation	248
5. Degree Of Success	248
6. Costs	248
7. Benefit Determination	251
8. Comments In Retrospect	251
9. Future Plans	251
F. Case History #6: Wang VS	252
1. The Company	252
2. EDP Environment	252
3. Productivity Problems/Selected Techniques	253
4. Technique Evaluation	253
5. Degree Of Success	255
6. Costs	255
7. Benefit Determination	255
8. Comments In Retrospect	256
9. Future Plans	256
G. Case History #7: SADT	257
1. The Company	257
2. EDP Environment	257
3. Productivity Problems/Selected Techniques	258
4. Technique Evaluation	259
5. Degree Of Success	259
6. Costs	261
7. Benefit Determination	262
8. Comments In Retrospect	263
9. Future Plans	263
H. Case History #8: Data Catalog II	263
1. The Company	263
2. EDP Environment	263
3. Productivity Problems/Selected Techniques	264
4. Technique Evaluation	264
5. Degree Of Success	266
6. Costs	266
7. Benefit Determination	266
8. Comments In Retrospect	267
9. Future Plans	268
I. Case History #9: PSL/PSA	268
1. The Company	268

	<u>Page</u>
2. EDP Environment	268
3. Productivity Problems/Selected Techniques	268
4. Technique Evaluation	270
5. Degree Of Success	270
6. Costs	270
7. Benefit Determination	270
8. Comments In Retrospect	272
9. Future Plans	273
J. Case History # 10: Universal Productivity Package (UPP)	273
1. The Company	273
2. EDP Environment	273
3. Productivity Problems/Selected Techniques	274
4. Technique Evaluation	274
5. Degree Of Success	274
6. Costs	276
7. Benefit Determination	276
8. Comments In Retrospect	277
9. Future Plans	277
VII MEASUREMENT AND EVALUATION STRATEGIES	279
A. Objectives In Measurement And Assessment	279
B. Uses Of Measurement And Assessment Methods	280
1. Evaluation Of The Software Product	280
2. Improving The Development Process	281
3. Improving The Maintenance Process	282
4. Assessing User Satisfaction	282
5. Assessing Organizational Effectiveness	283
C. Assessment As A Management Tool	283
D. Measurement As A Management Tool	285
1. The General Problem	285
2. Desirable Characteristics Of Measurements	290
3. Productivity Measurements Currently Used	292
4. Promising Measurement Approaches	302
5. Comparative Analysis Of Measurement Approaches	305
VIII RECOMMENDATIONS	311
A. Overview	311
B. Developing A Strategic Plan	313
C. Specific Strategic Recommendations	315
D. Tactical Recommendations	317
E. Recommendations For Small Organizations	321
APPENDIX A: INTERVIEW PROGRAM	322
APPENDIX B: BIBLIOGRAPHY	323
APPENDIX C: GLOSSARY	331

	<u>Page</u>
APPENDIX D: STRATEGY FOR A TRAINEE-RICH ORGANIZATION	349
A. The Problem	349
B. The Options	350
C. The Reason For The Problem	351
D. The Solution: A Trainee-Rich Personnel Strategy	353
E. Strategy Requirements	355
F. Organizational Impact	357
APPENDIX E: SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	359
APPENDIX F: QUESTIONNAIRE	397

IMPROVING THE PRODUCTIVITY OF SYSTEMS AND SOFTWARE IMPLEMENTATION

LIST OF EXHIBITS

			<u>Page</u>
II	-1	Disk Storage Cost/Performance Curve	6
	-2	Percent Of EDP Personnel Budget Devoted To Software Maintenance And Enhancement	8
	-3	Time Usage In Systems Development	10
	-4	The Productivity Pyramid	13
III	-1	Relative Importance Of EDP Problems	20
	-2	Distribution Of An Application Programmer's Time	23
	-3	How EDP Managers Define The Systems And Software Productivity Problem	27
	-4	Most Important Factors Inhibiting Productivity Improvement As Reported By Respondents	28
	-5	Needed Increase In Software Productivity As Reported By Respondents	31
	-6	Status Of Productivity Planning As Reported By EDP Managers	33
	-7	Responsibility For Productivity Improvement As Reported By EDP Managers	34
	-8	Areas Of Major Importance In Improving Software Productivity, As Reported By EDP Managers	35
	-9	Areas Of Major Importance In Improving Software Productivity, As Reported By Respondents	36
	-10	Areas Of Lesser Importance In Improving Software Productivity, As Reported By EDP Managers	38
	-11	Areas Of Lesser Importance In Improving Software Productivity, As Reported By Respondents	39
	-12	Action Taken To Improve Productivity, As Reported By EDP Managers	40
	-13	Relative Scope Of EDP Productivity Targets	47
	-14	Typical Project Workloading Curve	49
	-15	Impact Of Correcting Errors During Software Application System Life Cycle	51
	-16	The Management Nexus	53
	-17	The Need For Resource Management Of Software Development Projects	56
	-18	The Impact Of Design Methodologies On The Life Cycle	57
	-19	Case Study: The XYZ Organization	59
	-20	Relative Costs Over System Life	60

		<u>Page</u>
	-21 Ranges Of Productivity	62
	-22 Reported Enrolled Majors And Degrees Awarded In Computer Science, Data Processing And Related Fields: 1976-1977	67
	-23 Anticipated Personnel Costs And Shortage	69
	-24 Minimum Impact Of 20% Turnover	75
	-25 Distribution Of Analyst/Programmer Turnover As Reported By EDP Managers	76
	-26 Distribution Of Analyst/Programmer Turnover As Reported By EDP Managers By Company Size	78
	-27 EDP Analyst/Programmer Turnover Level, As Reported By On-Site Respondents	79
	-28 Most Important Motivators For Analyst/Programming Personnel, As Reported By EDP Managers	82
	-29 Other Non-Salary Incentives For Productivity Improvement, As Reported By EDP Managers	84
	-30 Single Most Successful Factor For Productivity Improvement, As Reported By EDP Managers	85
	-31 Single Least Successful Factor For Productivity Improvement, As Reported By EDP Managers	86
IV	-1 Motivating Factors In Moving From Stage To Stage In Productivity Improvement	98
	-2 Stages Of Productivity Improvement: Cost/Benefit Relationships, Stage 0	100
	-3 Stages Of Productivity Improvement: Cost/Benefit Relationships, Stage 1	102
	-4 Stages Of Productivity Improvement: Cost/Benefit Relationships, Stage 2	104
	-5 Stages Of Productivity Improvement: Cost/Benefit Relationships, Stage 3	105
	-6 Stages Of Productivity Improvement: Cost/Benefit Relationships, Stage 4	107
	-7 Characteristics Of Stage Zero: Chaos	110
	-8 Characteristics Of Stage One: Control	111
	-9 Characteristics Of Stage Two: Quality	113
	-10 Characteristics Of Stage Three: Efficiency	115
	-11 Characteristics Of Stage Four: Value	116
	-12 System And Software Productivity Self-Analysis Matrix	124
	-13 Productivity Self-Analysis Matrix Priority Progression	129
V	-1 The "Productivity Problem Space"	137
	-2 Solutions Related To Problem Space	138
	-3 Life Cycle Costs (LCC) For Major Projects	143
	-4 Applicability Of Selected Tools	145
	-5 Impact Of Selected Tools	146
	-6 LCC Improvement Versus Initial Cost For Selected Tools	149

		<u>Page</u>
	-7 LCC Improvement Versus Learning Curve For Selected Tools	150
	-8 LCC Base Data - Structured Tableau	153
	-9 LCC Base Data - PDL	155
	-10 Tool Profile: PWB/UNIX	164
	-11 Structured Programming Constructs	168
	-12 Warnier-Orr Diagrams	178
	-13 Tool Profile: SADT	183
	-14 Tool Profile: Structured Tableau	186
	-15 Tool Profile: PRIDE/Logik-ADF	194
	-16 Tool Profile: PSL/PSA	199
	-17 Tool Profile: PDL	202
	-18 Tool Profile: Raytheon's UPP	207
VI	-1 Company 5, Productivity Technique Evaluation: Optimizer III	249
	-2 Company 5, Productivity Technique Evaluation: Mark IV	250
	-3 Company 6, Productivity Technique Evaluation: Wang VS	254
	-4 Company 7, Productivity Technique Evaluation: SADT	260
	-5 Company 8, Productivity Technique Evaluation: Data Catalog II	265
	-6 Company 9, Productivity Technique Evaluation: PSL/PSA	271
	-7 Company 10, Productivity Technique Evaluation: Universal Productivity Package (UPP)	275
VII	-1 The Measurement Dilemma: Pages, Lines, Bytes	288
	-2 The Measurement Dilemma: Maintenance	289
	-3 Primary Measures Of Software Development Productivity, As Reported By EDP Managers	293
	-4 Lines Of Code - What To Count	298
	-5 "Normal" Rate And Variations Of L.O.C.	300
	-6 Rating Of Software Implementation Measurements	309
	-7 Ranking Of Productivity Measurements	310
E	-1 Total Rating For Self-Analysis, Activity-Oriented, Productivity Audit By Stage	361

I INTRODUCTION

I INTRODUCTION

- During the summer of 1979, INPUT began to discuss the subject of software productivity with a number of key clients to INPUT's Planning Service For Computer And Communications Users. The influence of those early discussions is evident in this report.
- A group of twelve charter sponsors, representing a cross-section of billion-dollar organizations in American enterprise, was assembled to guide and direct the study and to serve as a "living laboratory" for the analysis of the current practical state of the art in applications software development and implementation.
- In addition to in-depth interviews and analyses conducted at numerous staff levels (programmer/analyst through MIS Director) within these charter organizations, another 51 firms were visited and interviewed in person to determine their experiences with specific productivity approaches to software, and 32 more organizations discussed similar issues by telephone (see Appendix C), for a total of nearly 100 organizations and over 200 individuals furnishing the primary data for the study.
- Finally, a total of 1,300 mail surveys were conducted to determine statistically how widespread the problems are, how seriously they are affecting software development and maintenance, and what remedies are being applied to try to alleviate the situation.

- The results of this massive research effort have been condensed into the framework of this report, which is intended to put pragmatic boundaries around the "productivity gap" in an applications software context, and to lend perspective to the degree and type of interactions between contributing factors that must be considered in developing a software productivity improvement strategy.
- Management must not underestimate the seriousness of the problems outlined in succeeding chapters. It is INPUT's considered opinion that a software productivity improvement strategy lies at the root of what will be the MIS organization's total survival strategy for the eighties. It could significantly affect the course of the entire corporation as well.
- As organizations' information needs become more urgent and more complex, the demands on software production will increase beyond the range of tactical solutions.
- Hence the present study is presented in a framework that:
 - Begins with a discussion of parameters and factors causing the problem (Chapter III).
 - Proceeds with the development of a method and instrument to determine the nature and extent of the problem in each organization's particular context (Chapter IV).
 - Continues with a description and assessment of the various strategies, tactics, tools and aids available to address the problem (Chapter V), along with a number of "roadmaps" that have already been tried by various pioneering organizations, and an assessment of their effectiveness in meeting their goals (Chapter VI).

- Presents the pros and cons of various measurement techniques and their value in preparing and controlling a software productivity improvement plan (Chapter VII).
 - Ends with conclusions and recommendations for specific management strategies to deal with the overall software productivity improvement problem (Chapter VIII).
- A number of appendices furnish definitions and sources for further information, as well as practical details for implementing certain strategies.
 - Comments and questions concerning any aspect of the study are invited, and may be addressed to the nearest INPUT office, referencing this study by name.

II EXECUTIVE SUMMARY

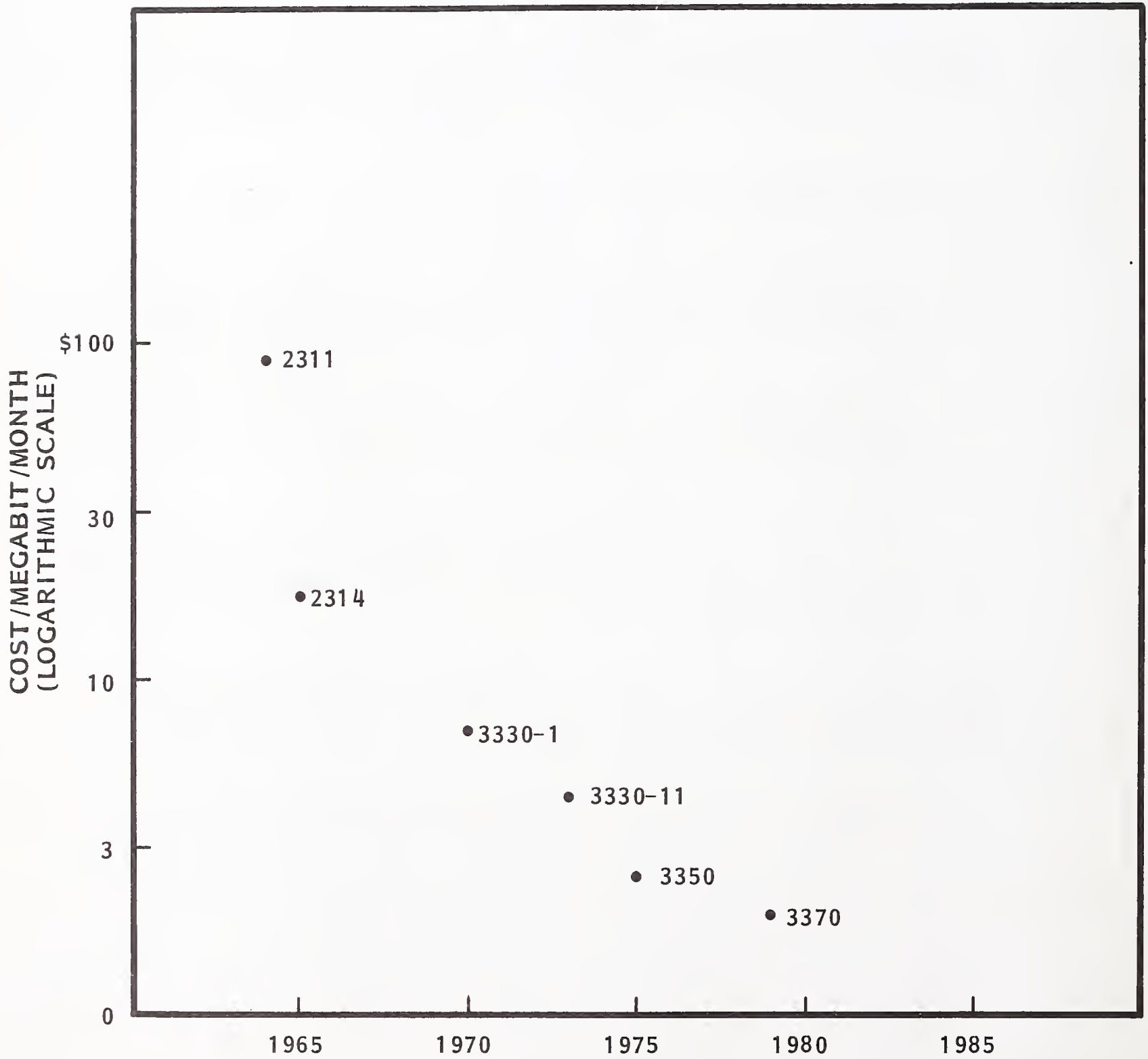
II EXECUTIVE SUMMARY

A. CURRENT "SOLUTIONS" DO NOT SOLVE PRODUCTIVITY PROBLEMS

- Most data processing organizations face some serious problems:
 - Data processing systems are growing more complex and expensive, and are taking longer to implement. Time and cost overruns are common.
 - Software maintenance consumes a larger share of the software dollar; users are dissatisfied with quality and reliability.
 - The backlog of unmet programming requests continues to grow, in spite of the fact that software-related expenses make up an increasing portion of DP budgets.
- These problems would have been even greater if significant improvements in technology had not masked their effects.
 - Capabilities and reliability of hardware have increased markedly.
 - Costs per performance unit have fallen spectacularly.
- As a result, an unchanged (or even decreased) hardware budget can buy much greater capacity, as shown in Exhibit II-1.

EXHIBIT II-1

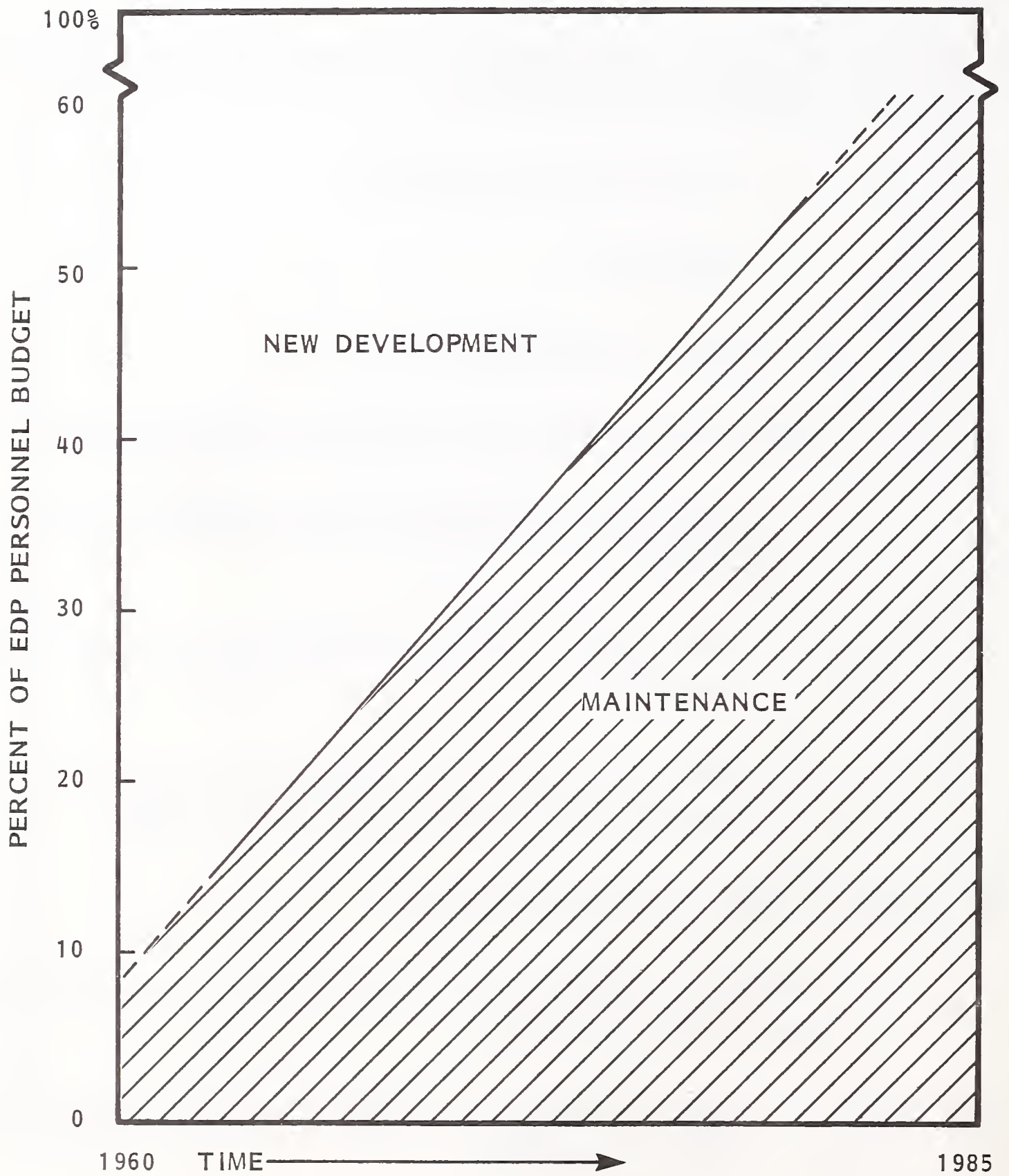
DISK STORAGE COST/PERFORMANCE CURVE



- However, software costs, especially maintenance costs, have risen dramatically over the last twenty years, as shown in Exhibit II-2. In contrast to hardware, improvements in software reliability and capability have been erratic, and can often be obtained by only a few installations.
- Current "solutions" usually fall into two general categories:
 - Attempts to copy hardware successes by using analogous technical improvements; for example:
 - On-line program development.
 - Coding aids.
 - Design methodologies.
 - Organizational changes, sometimes merely cosmetic, including:
 - Charge-back arrangements that don't allow the use of alternate suppliers.
 - Users seeking control over "their" systems via DDP or decentralization.
 - Self-serving performance measurements designed by the DP department to show that its performance is adequate.
- Neither approach has been very successful.
 - The technical improvements are rarely part of a thorough problem analysis and overall strategy.
 - Hence, they usually focus on only a small part of personnel time (i.e., the coding phase).

EXHIBIT II-2

PERCENT OF EDP PERSONNEL BUDGET DEVOTED
TO SOFTWARE MAINTENANCE AND ENHANCEMENT



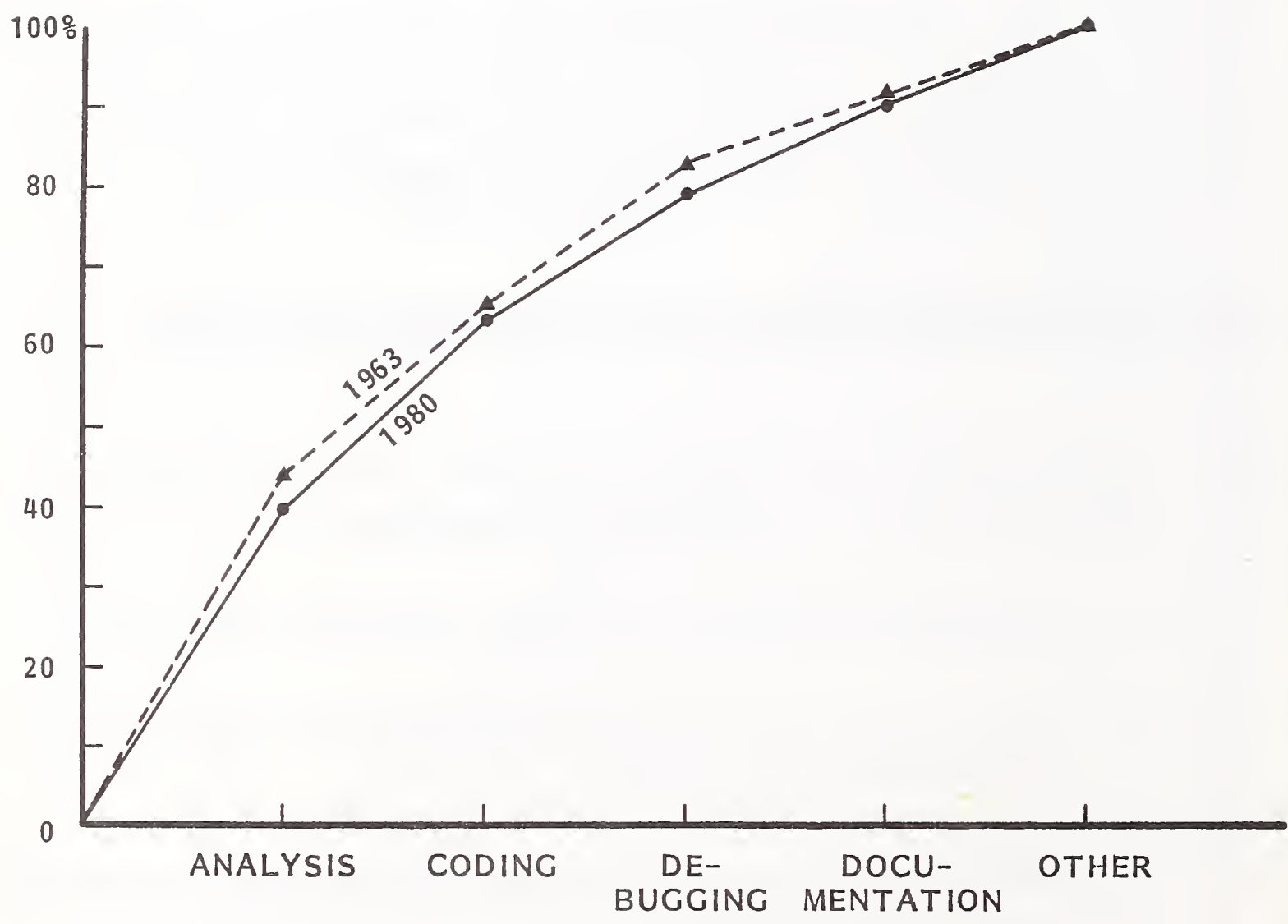
- By ignoring the organizational situation and a suitable integrating strategy, technical aids may actually be counter-productive.
- In spite of much publicity over productivity aids and approaches, they have not had much impact.
- In fact, the systems development process has been remarkably unchanged for almost twenty years, as shown in Exhibit II-3.
- The organizational approach (DDP, etc.) has also not been very successful at delivering the promised improvements, while the technical and managerial complications associated with this approach are often much larger and more severe than anticipated.

B. CURRENT PERCEPTIONS OF THE PROBLEM ARE TOO LIMITED

- INPUT's survey research for this report has found that productivity is considered primarily a schedule-overrun/backlog problem.
 - Increasing costs and cost overruns are considered secondary issues.
 - Software quality, user expectations and personnel issues were rarely cited as contributing problems.
- DP management believes that solutions to productivity are largely dependent on tools and tactical issues such as:
 - Personnel factors (motivation and skills).
 - Techniques (measurement and requirements definition).

EXHIBIT II-3

TIME USAGE IN SYSTEMS DEVELOPMENT



NOTE: CURVE IS CUMULATIVE

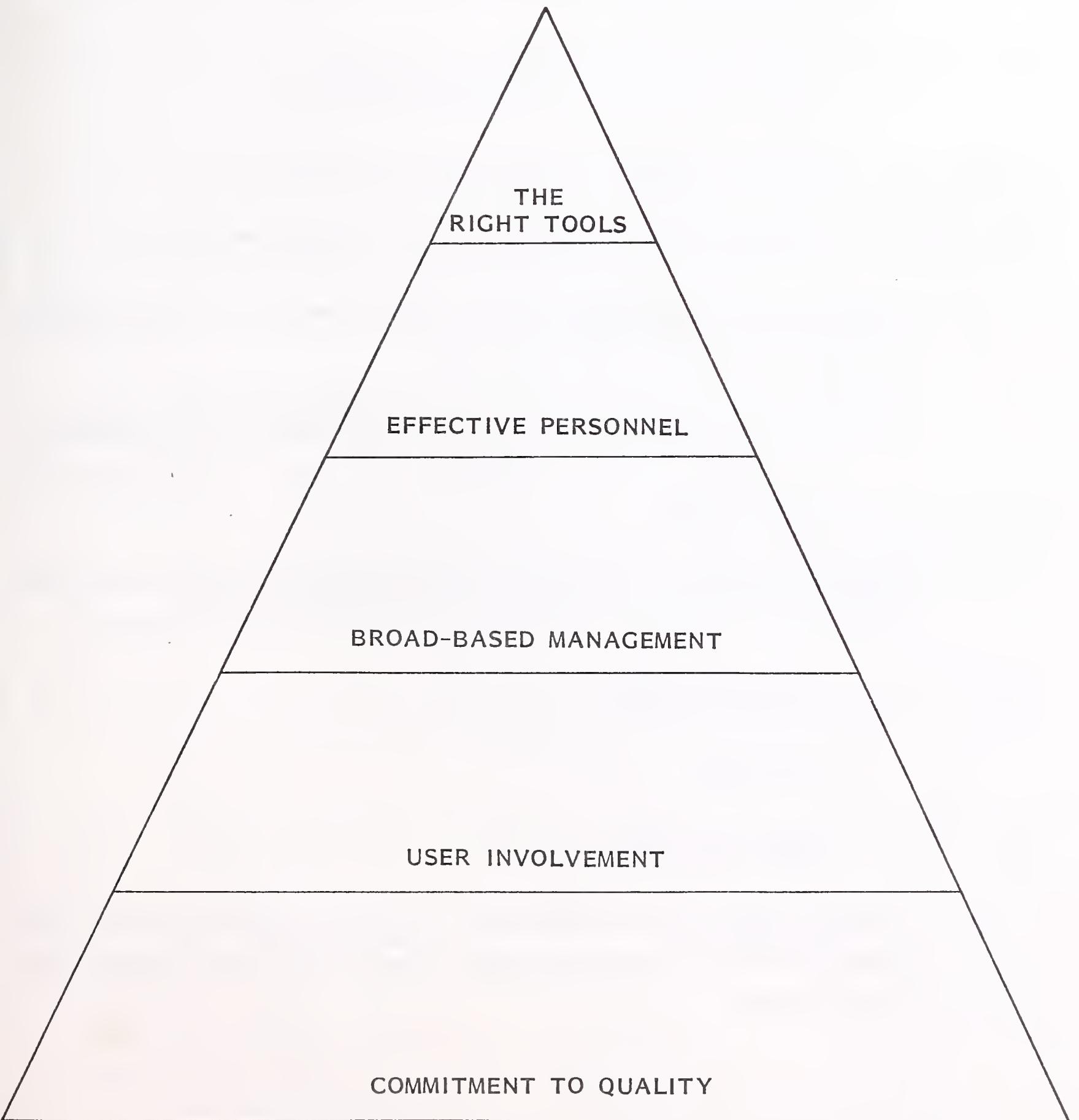
- Scheduling.
- DP management feels that productivity is not primarily related to more general issues such as:
 - The role of the end user.
 - Planning and management skills.
 - The corporate environment and policies.
- Actions taken to improve productivity include:
 - Supplying better tools.
 - Better planning and training.
 - More user involvement.
- However, in no case was any action undertaken by more than one-quarter of those surveyed. In only a few cases the actions were part of a thought-out plan that promised to be effective.
- The picture of the typical DP department that has emerged from INPUT's research is that of an inward-looking, technically focused organization that views improved productivity as a marginal increase in the amount of coding done per day.
 - This approach will prevent DP from fulfilling more than a fraction of its potential for the organization.
 - In addition, such a limited approach almost guarantees the ultimate failure, and eventual replacement, of present DP management.

C. A PRODUCTIVITY STRATEGY IS REQUIRED

- DP productivity is in fact a company-wide issue. However, few people perceive it as such, including most DP management.
 - DP's aim is (or should be) to support, improve and integrate functions throughout the entire organization.
 - If DP retreats by attempting to improve only the functions under its direct control, then DP management will find that:
 - It has a much smaller number of issues under its influence.
 - Even in these, it cannot accomplish as much.
- A productivity strategy requires the following components:
 - A commitment to quality.
 - User involvement.
 - Broad-based management.
 - Effective personnel.
 - The right tools.
- The order given above is important because the components are interrelated, as shown graphically in Exhibit II-4.
- The most effective strategy is one that focuses on building productivity from the base up.

EXHIBIT II-4

THE PRODUCTIVITY PYRAMID



- A commitment to quality is the foundation for all else.
 - The quality of the system, especially its architectural stability, will determine its ultimate success or failure.
- User involvement is essential. The optimum approach is to move to a position where users are:
 - Involved in systems development and operation.
 - Informed on what DP can and cannot do for them.
 - Aware of how their needs fit into larger company requirements.
- Broad-based EDP management puts the needs of users and top management on an equal footing with running the DP shop.
 - One of the biggest opportunities (and challenges) is for DP management to educate both top management and users in non-technical DP fundamentals.
- Effective personnel are the key to improving productivity (narrowly defined as production rate or quantity of software). Strategies here should focus on:
 - Employee retention.
 - Motivation.
 - Skills improvement.
- The right tools are the best means to achieving "microproductivity," but contribute little to "macroproductivity" unless the other layers of the pyramid are all in place.

D. THERE IS NO SINGLE STRATEGY FOR ALL ORGANIZATIONS

- There is no definitive strategy useful to all organizations.
 - The stage of development of a particular organization will greatly influence which approaches are appropriate.
 - The individual characteristics of an organization determine which approaches are feasible.
- Successful strategies take time to implement. The planning and implementation of successful strategies can, and often should, take years before all the major pieces are in place. Of course, much work can be done immediately, but such areas as user involvement can take a long time.

E. STAGES OF PRODUCTIVITY DEVELOPMENT: A KEY CONCEPT

- Key to a successful software productivity strategy is the choice of individual productivity initiatives that are consistent with a data processing organization's current stage of development.
 - If a particular productivity initiative is too far ahead or behind what the DP department is doing in other areas, not only may the initiative fail, but it may run a high risk of being counterproductive.
- From the standpoint of productivity, there are five definable stages in DP development.
 - Stage 0: Chaos.
 - Stage 1: Control.

- Stage 2: Quality.
 - Stage 3: Efficiency.
 - Stage 4: Value.
- In each stage, different productivity strategies are called for. In the later stages, the productivity initiatives are often more complex and difficult, but the payoff can be very high.

F. SELF-ASSESSMENT

- This report provides extensive materials to enable organizations to undertake a self-analysis in two critical areas:
 - Their stage of productivity development.
 - Whether they are using the instruments of productivity appropriate for their particular stage of development.
- Determining the stage of productivity is important because:
 - Some components will be too far ahead or behind the others, and corrective action is indicated to bring them into balance.
 - The next stage of development for each component, as well as its relative importance, can be planned as part of an integrated strategy.
- Analyzing the appropriateness of individual productivity instruments is equally important.

- INPUT has provided a self-audit point score for each productivity approach.
- In some cases the score for a single approach will vary, depending on the stage of development.
 - . This reflects the situation where, for example, sophisticated planning would be appropriate in later stages but would be counterproductive in the initial "chaos" stage.
- This point score approach gives organizations the ability to judge the appropriateness and potential payoff of particular productivity initiatives.

III DIMENSIONS OF THE PRODUCTIVITY PROBLEM

III DIMENSIONS OF THE PRODUCTIVITY PROBLEM

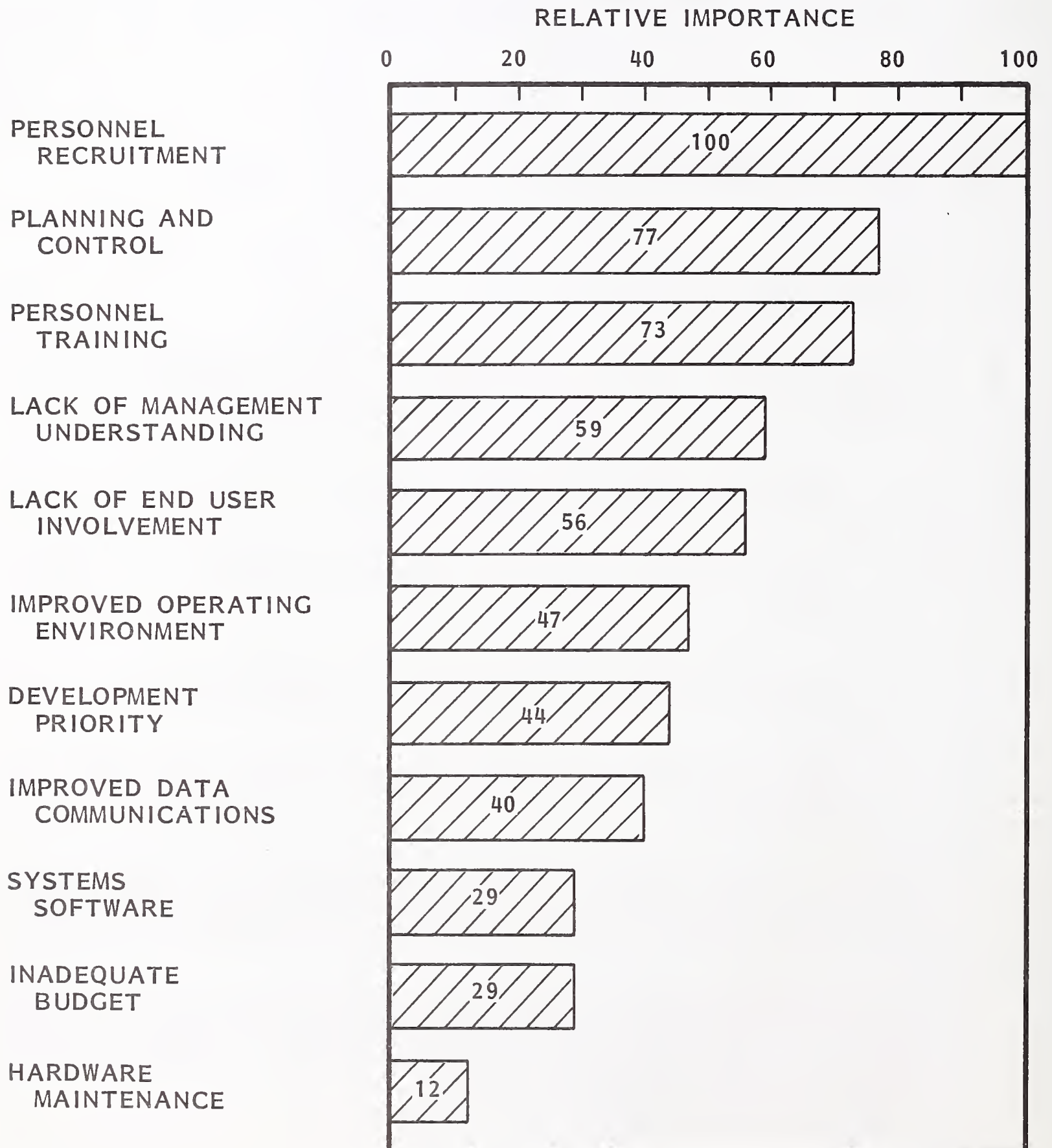
A. PERCEPTION AND DEFINITION

I. WHERE DOES THE PROBLEM LIE?

- To all those concerned with data processing - whether as suppliers or users of services - the following litany of complaints is all too familiar:
 - New systems are often late and over budget.
 - Systems are unreliable, with never-ending maintenance "enhancements."
 - Users are ignorant and irresponsible (from the EDP standpoint).
 - The "technicians" are unresponsive and arrogant (from the user's standpoint).
- Is there any basis for these complaints, or is it simply a case of finger-pointing in the heat of pressures from the general competitive and economic situation?
- In the 1980 survey INPUT conducted of more than 900 EDP organizations, recruitment and training of personnel were the first and third most important problems cited, as shown in Exhibit III-1.

EXHIBIT III-1

RELATIVE IMPORTANCE OF EDP PROBLEMS



USER PANEL; NUMBER OF RESPONSES = 912

- In the same survey, management-related problems were also of high importance, including problems in:
 - Planning and control (ranked second in importance).
 - Lack of upper management understanding (ranked fourth).
 - Lack of end user involvement (fifth).
 - Setting priorities for development (eighth).
- Technical problems per se did not rank very high on the problem list, including only:
 - Improved operating environment (sixth).
 - Improved data communications (seventh).
 - Systems software (ninth).
 - Hardware maintenance (eleventh).
- Those in EDP know very well that many of the technical problems they face are serious and demanding. Rankings such as those above indicate that non-technical problems are also very serious indeed.
- It would be reasonable to conclude that, if solutions are not found within or by the EDP organization, other groups will take the initiative to impose "solutions."

2. HISTORICAL PERSPECTIVE

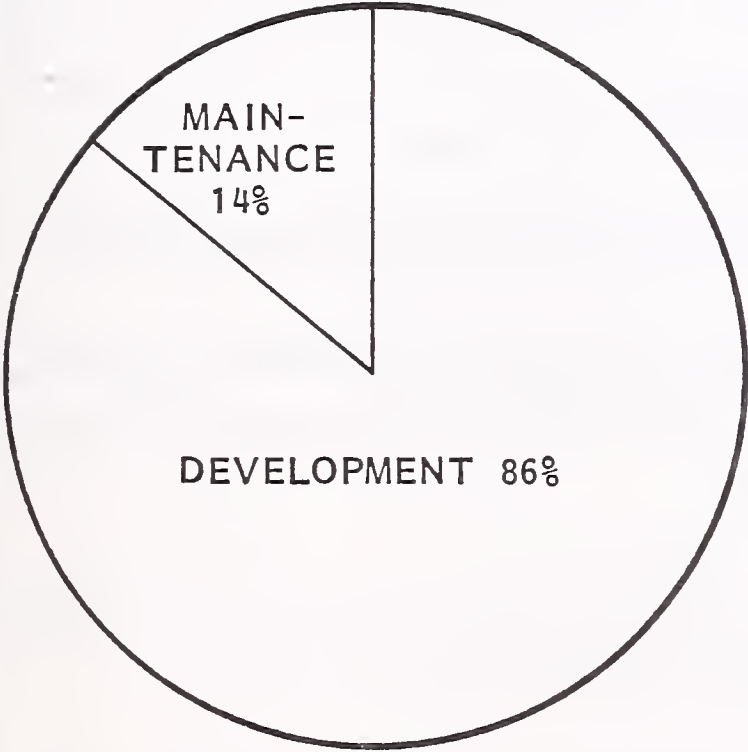
- Many of the problems in data processing appear familiar, perhaps because the essentials of data processing have not changed since the days of the second-generation machines of twenty years ago.

- Hardware is now larger, faster and cheaper, but hardly anything else has changed in two decades.
- Not only has the basic approach to data processing changed little as far as the analysis-programming-testing cycle goes, but even the proportions of time spent on the different activities still obey the venerable rule of thumb: 40% analysis, 40% coding and debugging, and 20% everything else.
- For a profession that likes to think of itself as constantly changing and improving, it is disturbing that the change has only occurred in terms of hardware.
- It would be one thing if the traditional software development approach had been so successful that there was no point in switching from a winning strategy. However, the unforeseen and apparently inexorable increase in the time devoted to maintenance, and maintenance under the name of enhancement, effectively discredits the "system" that gave rise to it, as illustrated in Exhibit III-2.
- Not only does maintenance represent a serious drain on an organization's financial resources, but more critically:
 - It often represents systems that are not working correctly and are not serving users' needs.
 - For the typical EDP organization, it represents a wasteland where the rawest and worst programmers are detailed.
- Ironically, the better job programmers do in maintenance, the greater the temptation to keep them there. Frequently their only hope is to threaten to leave.

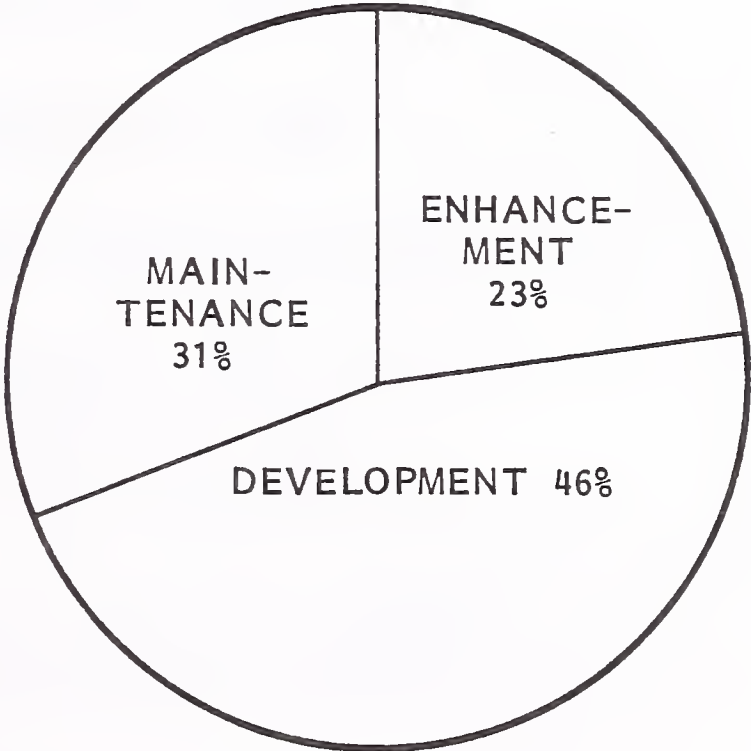
EXHIBIT III-2

DISTRIBUTION OF AN
APPLICATION PROGRAMMER'S TIME

1963



1980



- Maintenance thus represents a drain on an EDP organization's vitality, a constant threat to its morale, and a management albatross of major proportions.

3. CURRENT "SOLUTIONS"

- Most current software "solutions" focus on repeating the unparalleled success of computer hardware, apparently brought about by the use of engineering tools and aids. Touted among these have been:
 - On-line development, one terminal per programmer.
 - Software development aids (e.g., COBOL preprocessors).
 - Design methodologies (e.g., HIPO).
- These techniques have been described as the tools for a "software factory."
 - In theory, the workers in a software factory will have their jobs broken down and automated to such a degree that each person will only have to know and do a very few things.
- INPUT has not found much evidence that "software factories" are sweeping the country, or even that they represent a significant step forward.
- However, the use of software factory tools has now received enough publicity to become respectable.
 - Many EDP departments have, as a consequence, accumulated a grab bag of "productivity" tools that they believe (as a matter of faith) will make things better.

- At least the productivity tool missionaries and their converts are trying. More common in the business data processing world are the organizations that are attempting virtually no real improvements.
- For this larger group of companies, the "solutions" they attempt either backfire or amount to a merely cosmetic change.
 - An EDP department proposes that user departments ought to carry "their own" data processing costs in their own budgets or, in a more sophisticated form, the EDP group sets up a complicated "charge-back" mechanism, without allowing users the option of buying outside services.
 - Alternatively, "measurements" of the EDP operations section are instituted to show that the users' demands are unrealistic, that uptime exceeds 98% of available time, that "average" transaction response time is less than three seconds, and that 95% of the reported system bugs are responded to within 24 hours. More sophisticated measurers may also be able to show an improvement trend in at least one of these indicators over the last three months.
 - What is not reported is that uptime is calculated on a base that does not include "scheduled" downtime, that certain transactions or certain times of the day regularly produce response times in the multiminute range, and that the cost of repairing bugs amounts to almost one-third of the entire EDP personnel budget.
 - User departments, in turn, ask to be given "real" control over "their" systems, occasionally establishing a mini-EDP operation that is only slightly less adequate than what went on before.
 - Reorganization is a game that all can play, usually starting or ending with, "Bring me the head of ..."

B. PRODUCTIVITY ISSUES

I. SYMPTOMS OF THE PROBLEM

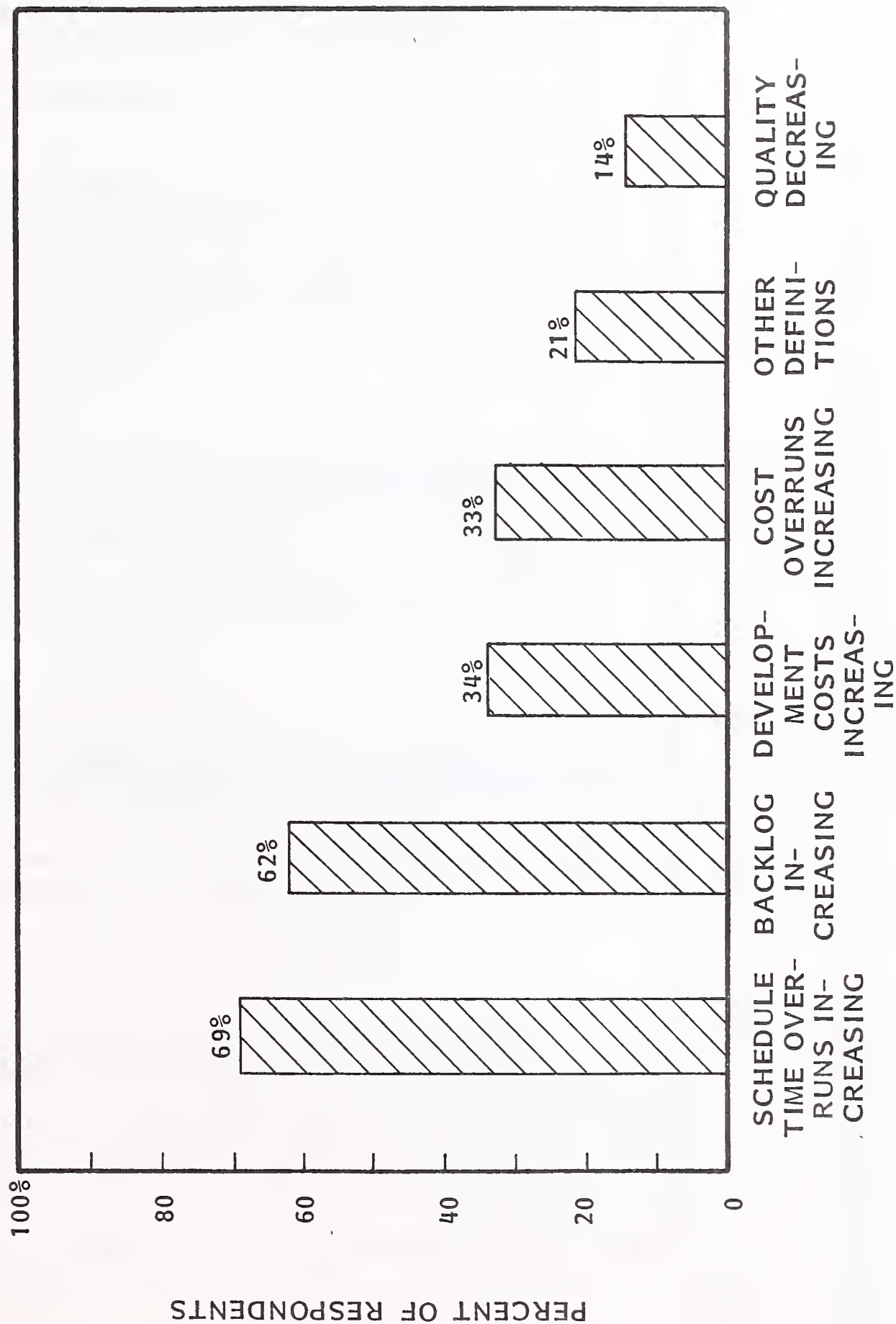
- Exhibit III-3 shows that when data processing managers were asked to define the composition of their systems and software productivity problems, they most often cited:
 - Increasing time overruns (69%).
 - Increasing work backlogs (62%).
- Far behind came:
 - Increasing development costs (34%).
 - Increasing cost overruns (33%).
- Besides confirming that time is more valuable than money, the responses were very interesting in that:
 - Decreasing quality was not considered a significant problem (named by only 14%).
 - Personnel issues were not seen as productivity problems, even though they were the number one general EDP problem!
 - Unrealistic user expectations were also not cited.

2. INHIBITING FACTORS

- Exhibit III-4 shows what EDP personnel see as the most important factors inhibiting productivity improvement. In order of importance, they are inadequacies in:

EXHIBIT III-3

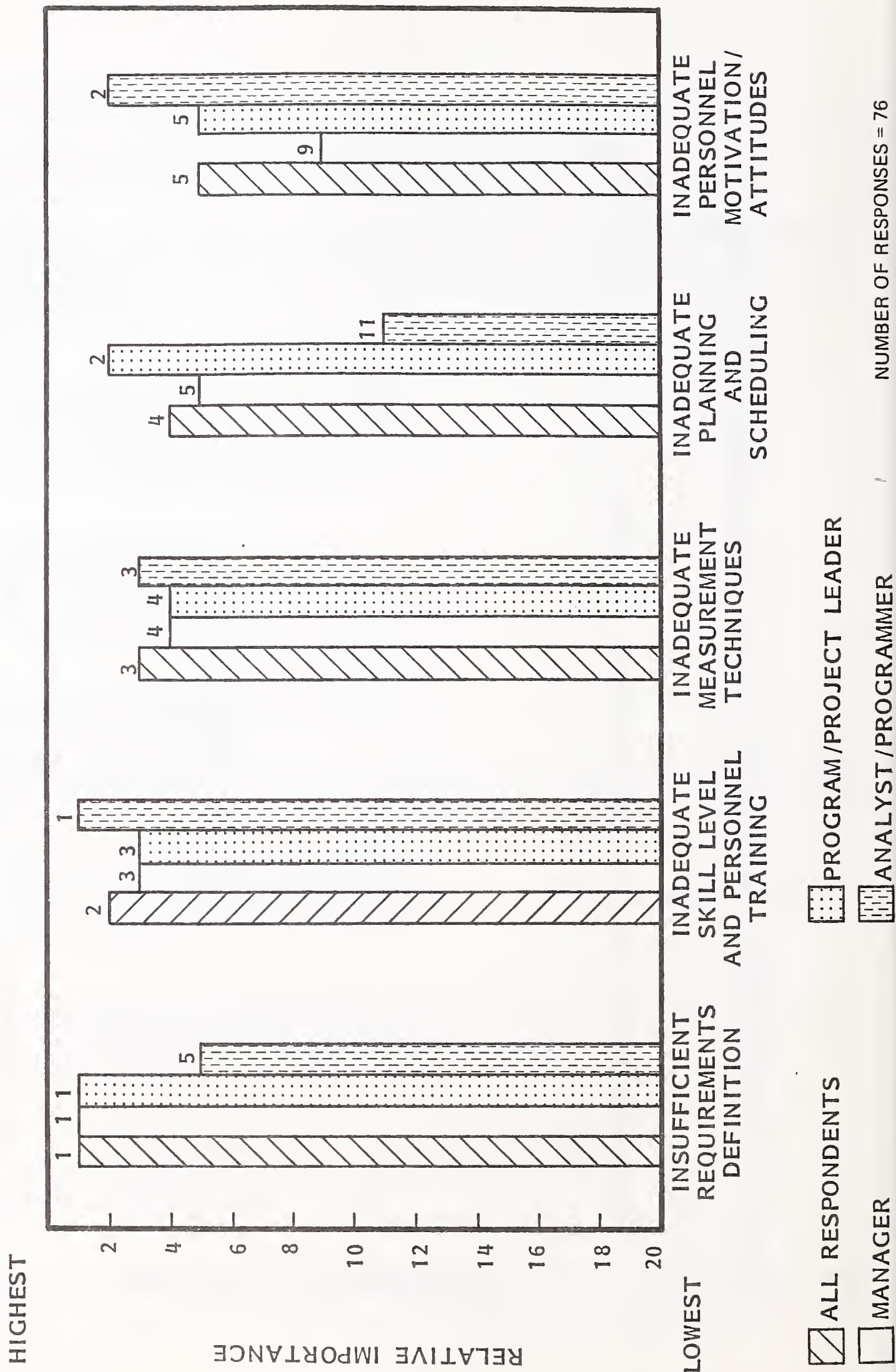
HOW EDP MANAGERS DEFINE THE SYSTEMS AND SOFTWARE PRODUCTIVITY PROBLEM



SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 262

EXHIBIT III-4

MOST IMPORTANT FACTORS INHIBITING PRODUCTIVITY IMPROVEMENT AS REPORTED BY RESPONDENTS



- Requirements definition.
 - Skill levels/training.
 - Measurement techniques.
 - Planning and scheduling.
 - Personnel motivation and attitudes.
- Of these factors:
 - Two are personnel-related (second and fifth).
 - Two are techniques (first and third).
 - One is a management issue (fourth).
 - These inhibiting factors are somewhat broader-based than the symptoms described earlier, but not nearly so comprehensive as the major overall problems described in Section A of this chapter.
 - More importantly, there is still an emphasis on tools and tactical issues; only planning (ranked fourth) is a general issue.
 - There is general agreement for the most part on the relative importance of the various inhibiting factors, especially concerning skills and measurement techniques (which also happen to be the most specific and the easiest to "do something about"). However, the differences, which show the insularity of personnel at different job levels, are quite disturbing.

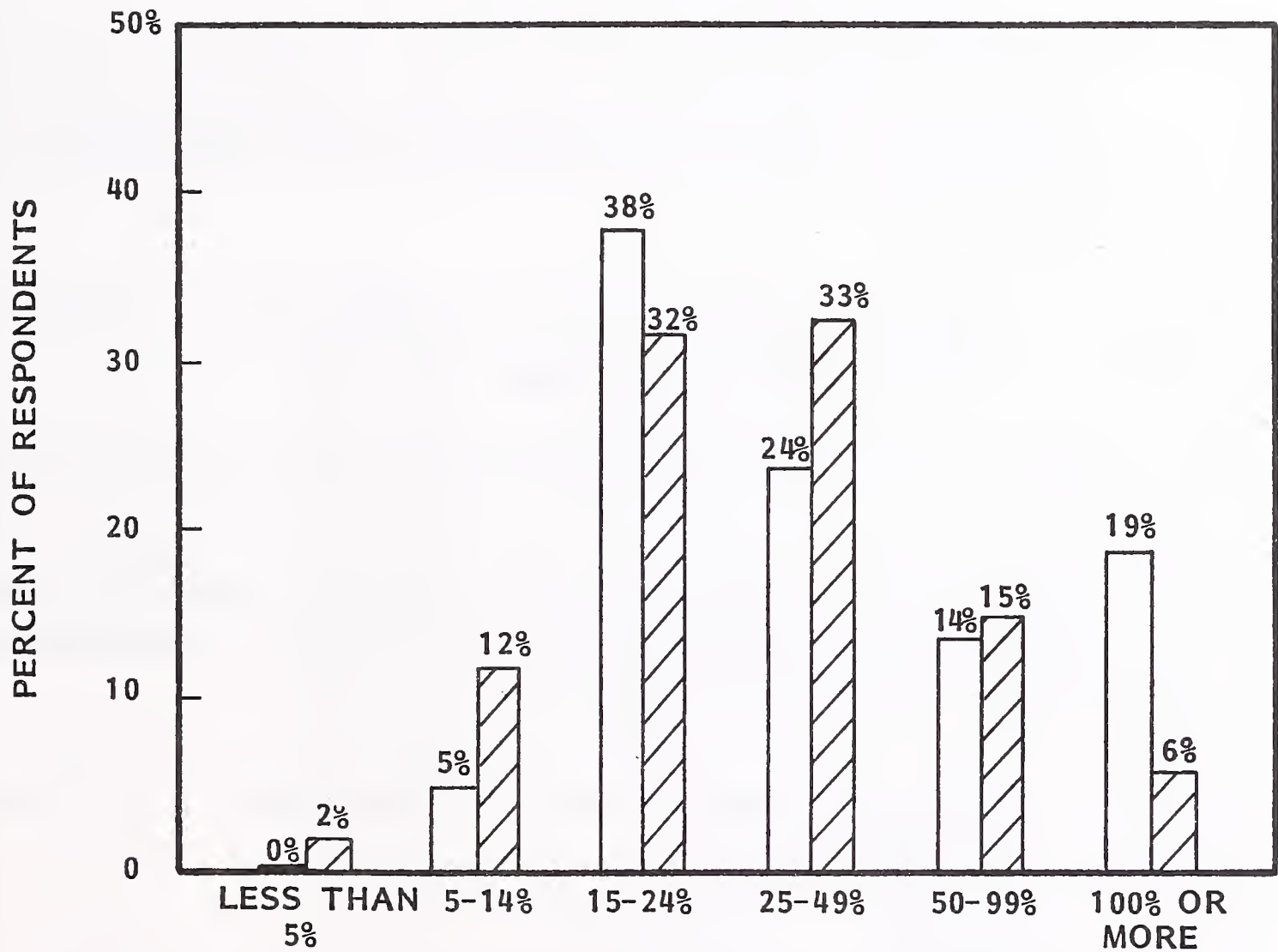
- Programmer/analysts see far less evidence of inadequate requirements definition (which is their job) than do others (who see it as the number one problem).
- Project leaders see inadequate planning as the second most important problem. (They, after all, are caught in the middle and probably have to take the heat for much bad planning.) Meanwhile, programmer/analysts are blissfully unaware.
- Programmer/analysts see inadequate motivation as the second most important problem (since they have to bear the brunt of "demotivational" policies). Managers, whose responsibility it is to motivate their staff, appear to believe that things are basically all right.
- At the least, there is a definite communication problem. Most likely, there is a general denial of responsibility.
 - If people with similar backgrounds and knowledge, who are in constant contact with each other, cannot agree on what is wrong, how can they jointly deal with the problems or effectively work with external groups in solving them?

3. GOALS AND MECHANISMS

- Managers want to see a significant improvement in "productivity," loosely defined as the amount of work that can be completed in a given period of time. About one-third believe that an improvement of 15-25% is needed, and almost as many believe an increase of 25-49% is needed (Exhibit III-5). About one-fourth want improvement of 50% or more. These are ambitious goals.
- In order to accomplish this, all parties must first agree that there is a problem.

EXHIBIT III-5

NEEDED INCREASE IN SOFTWARE PRODUCTIVITY AS REPORTED BY RESPONDENTS



- ☐ NUMBER OF ON-SITE RESPONSES = 21
- ☒ NUMBER OF MAIL RESPONSES = 284

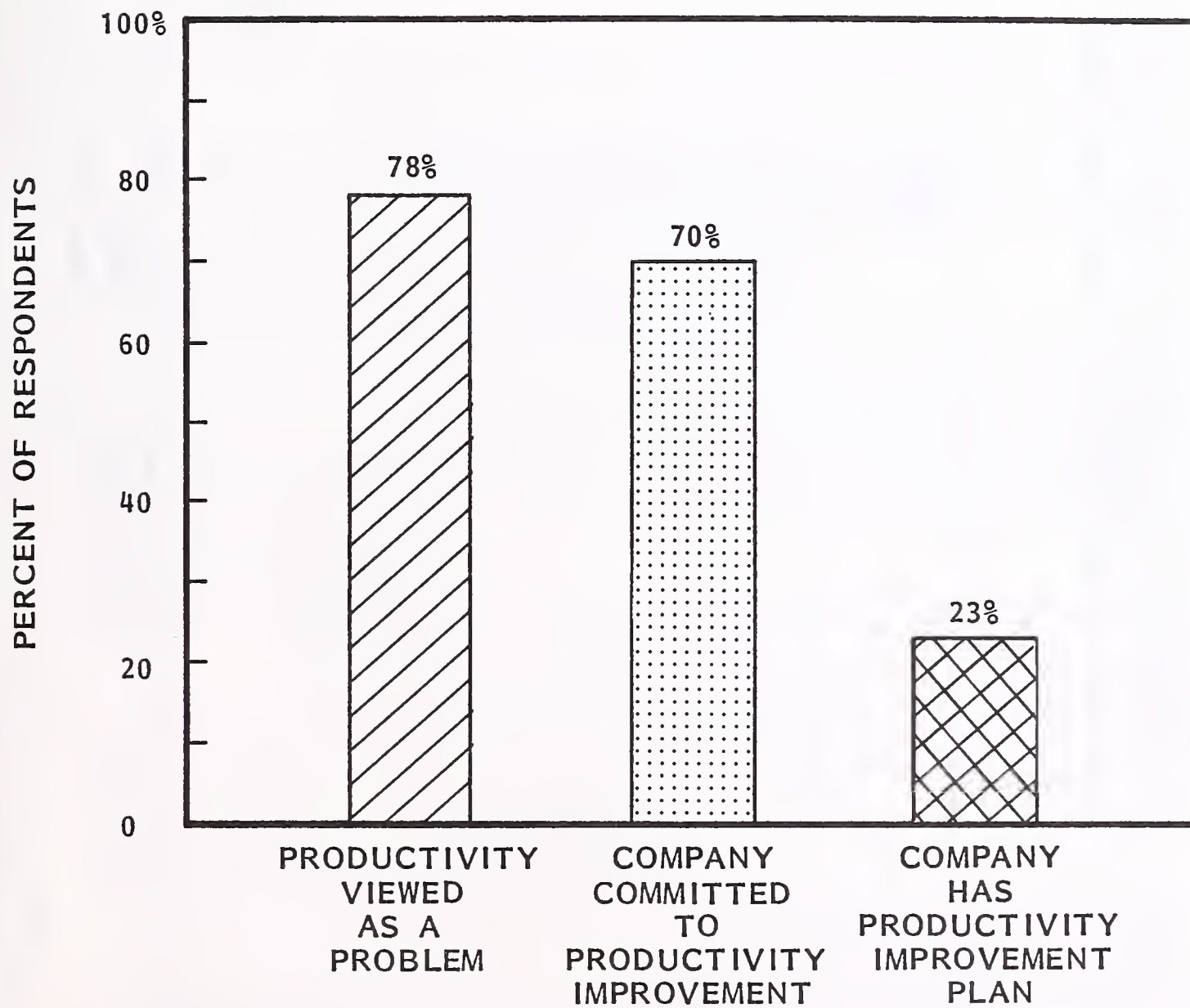
- Over three-fourths of the firms surveyed view productivity as a problem, as shown in Exhibit III-6. Almost as many have made a corporate commitment to improving the situation.
 - However, less than one-fourth have a productivity plan.
 - There is clearly some difficulty in moving from the "general talk" stage even to the planning stage.
- Responsibility for productivity improvement is assigned to a specific person or group in only 18% of companies surveyed (33% of EDP departments), as shown in Exhibit III-7. Otherwise, the responsible entity is "no one", "everyone" or "each manager."
 - This vague designation shows a lack of direction. It may very well represent a lack of commitment.

4. HOW THE EDP DEPARTMENT BELIEVES GOALS SHOULD BE ACHIEVED

- In ranking factors necessary to achieve productivity goals, EDP department personnel were more realistic in their clustering of areas believed important, as shown in Exhibit III-8. Of the top six:
 - Three were personnel related: motivation (first), training (third), and recruitment (sixth).
 - Two were related to management: motivation (second) and interest (fifth).
 - One was related to tools (fourth).
- The differences between the priorities of different personnel categories, shown in Exhibit III-9, were generally not striking, except that:

EXHIBIT III-6

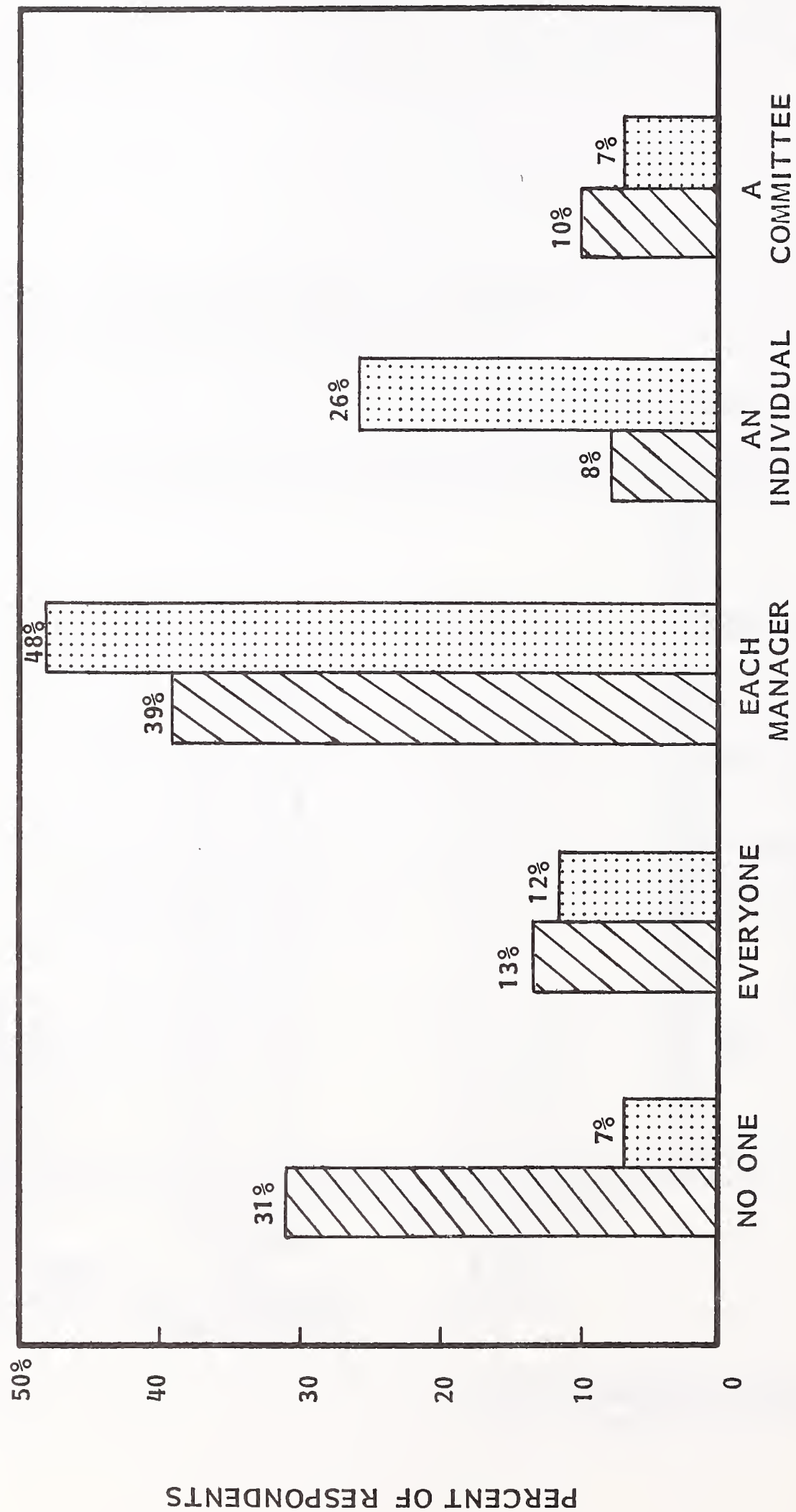
STATUS OF PRODUCTIVITY PLANNING AS REPORTED BY EDP MANAGERS



SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 300

EXHIBIT III-7

RESPONSIBILITY FOR PRODUCTIVITY IMPROVEMENT AS REPORTED BY EDP MANAGERS



COMPANY-WIDE
WITHIN EDP

SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 300

AREAS OF MAJOR IMPORTANCE IN IMPROVING SOFTWARE PRODUCTIVITY,
AS REPORTED BY EDP MANAGERS

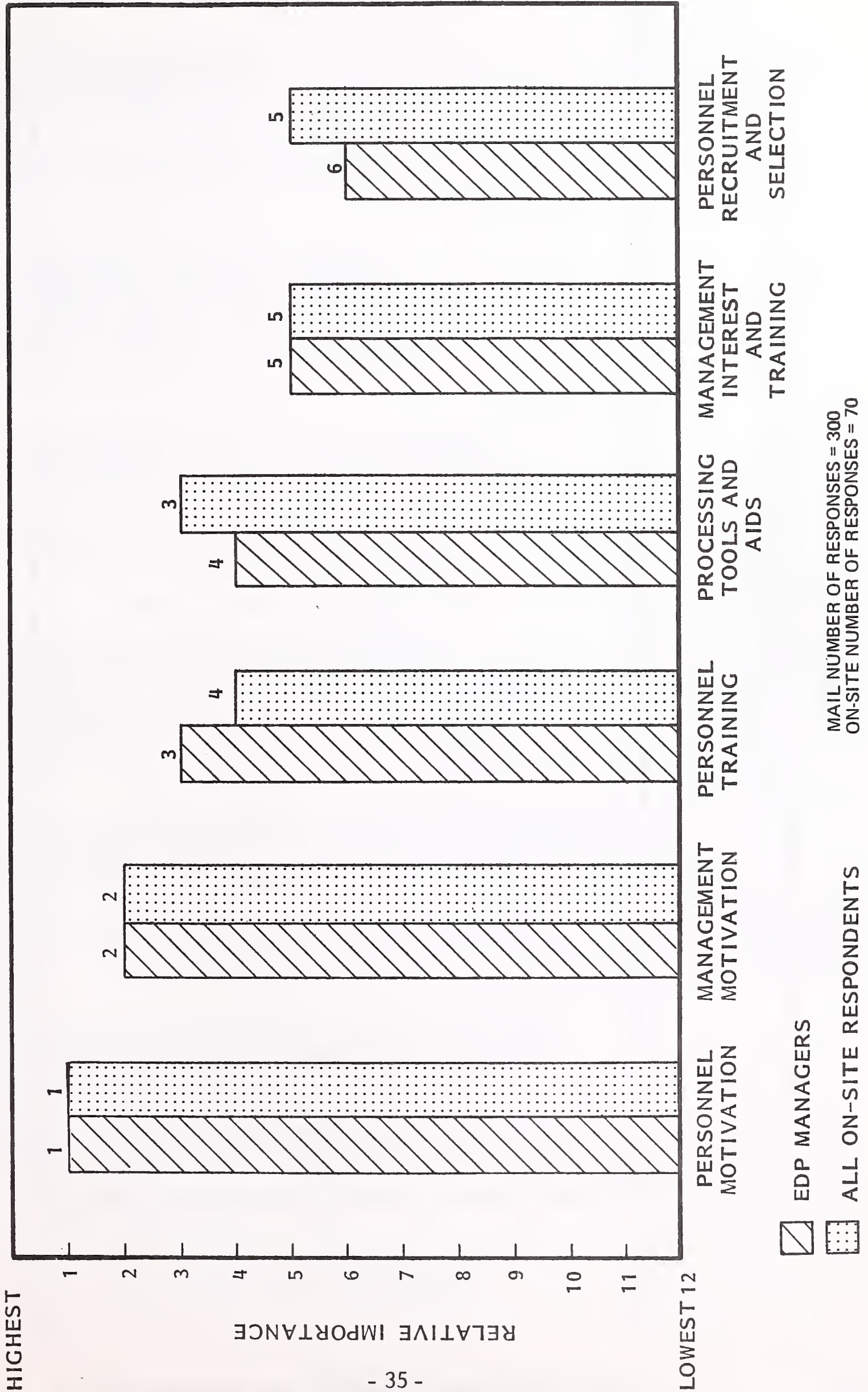
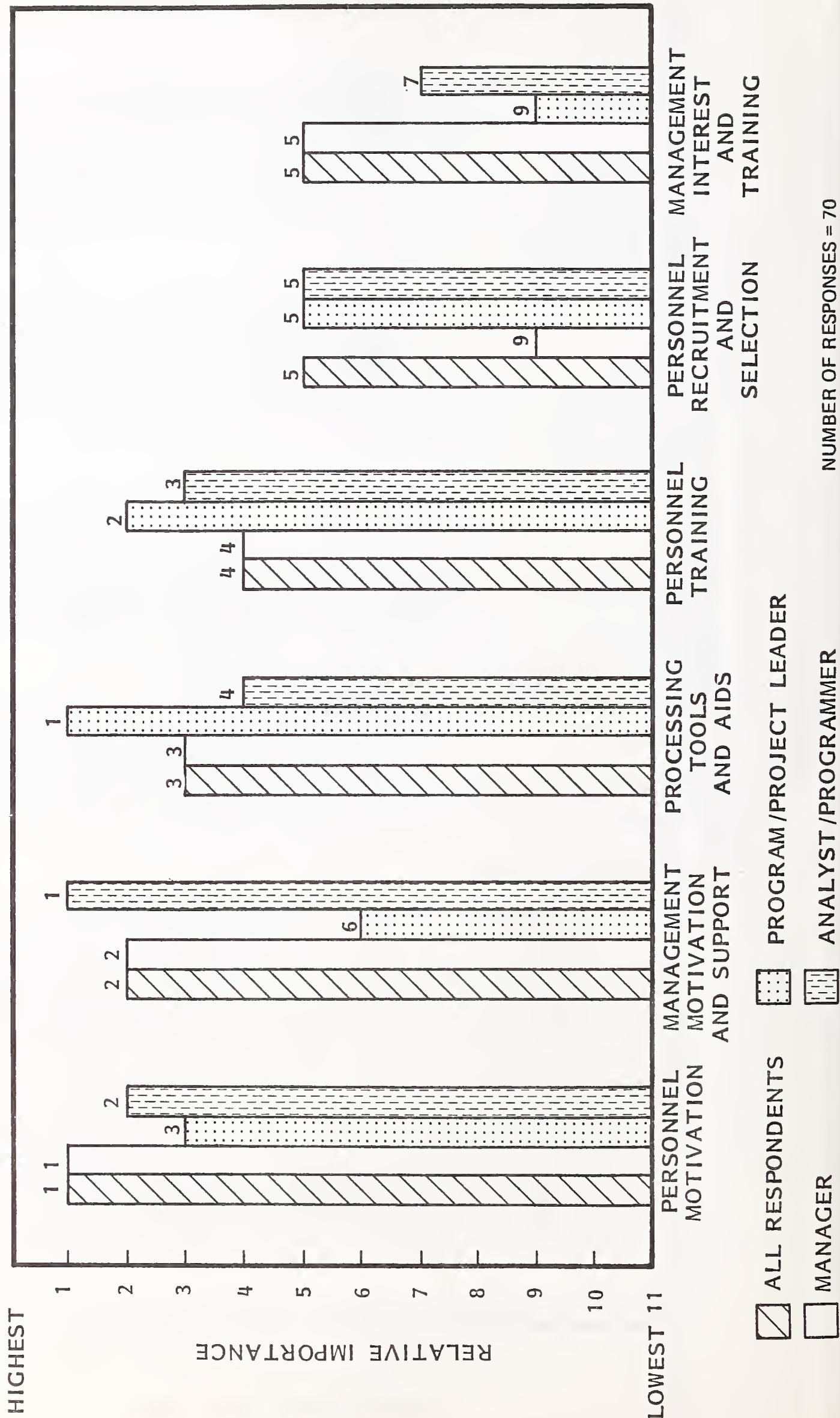


EXHIBIT III-9
AREAS OF MAJOR IMPORTANCE IN IMPROVING SOFTWARE PRODUCTIVITY,
AS REPORTED BY RESPONDENTS



- Project leaders put somewhat less emphasis on management motivation and interest, but somewhat more on tools, than other respondents.
- Managers put less emphasis on personnel recruitment. (Denial of responsibility?)
- Respondents generally felt that the broad-based issues were of less value in achieving goals, as shown in Exhibit III-10.
 - Role of the end user.
 - Planning tools.
 - Management skills.
 - Corporate environment and policies.
- Exhibit III-11 shows that there was not much difference in responses by level, except that:
 - Programmer/analysts felt that the end user role was somewhat more important (probably because they were on the front line).
 - Project leaders felt that environmental factors were much more important. Since the groups above and below them had agreed on these factors' lack of importance, this may be the tangible but impersonal factor on which to lay blame for projects that are late.

5. CURRENT ACTIONS TO IMPROVE PRODUCTIVITY

- What little is actually being done is consistent with the goals and beliefs described above (see Exhibit III-12). Of the six actions undertaken by 10% or more of respondents:

EXHIBIT III-10

AREAS OF LESSER IMPORTANCE IN IMPROVING SOFTWARE
PRODUCTIVITY AS REPORTED BY EDP MANAGERS

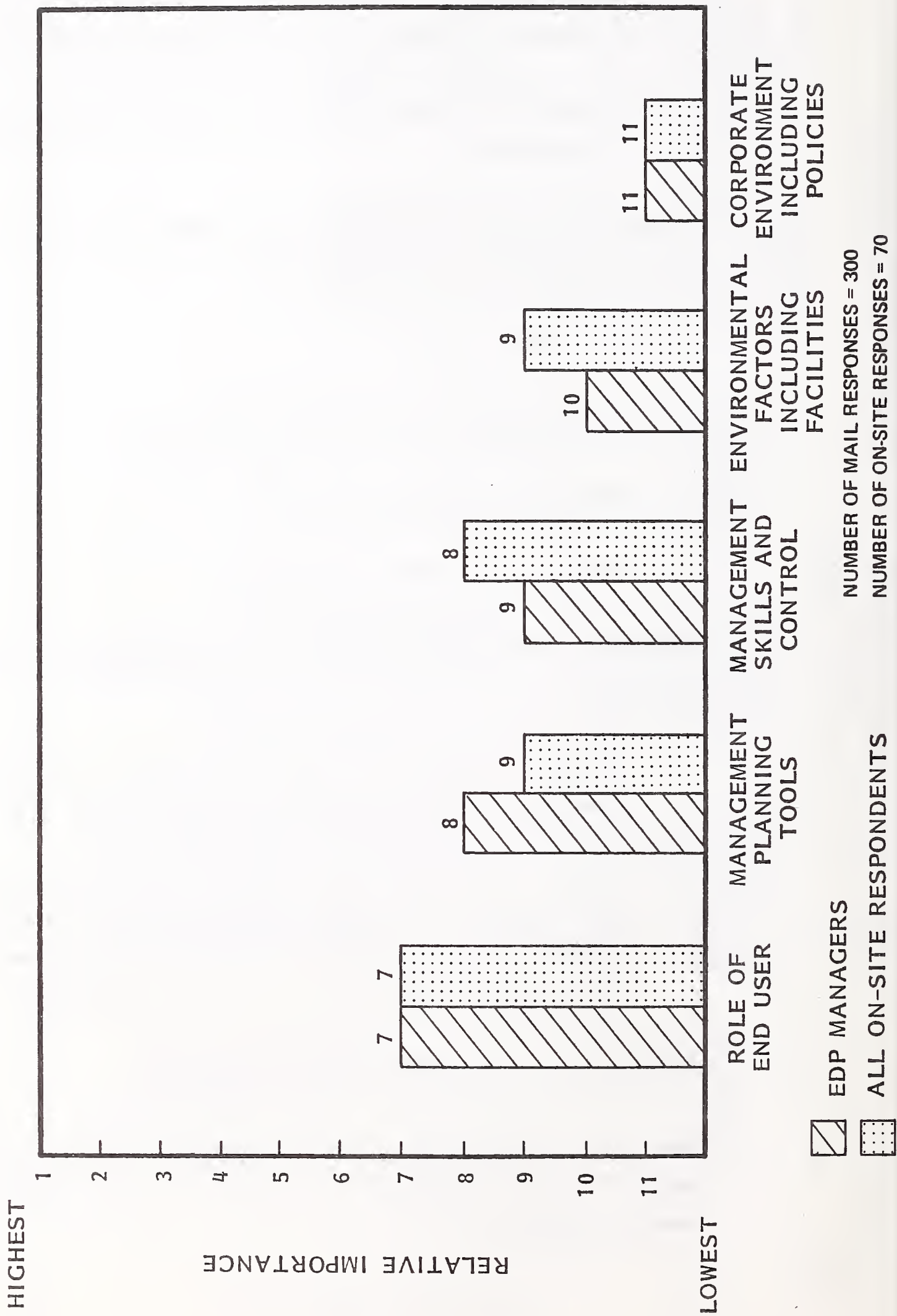


EXHIBIT III-11

AREAS OF LESSER IMPORTANCE IN IMPROVING SOFTWARE PRODUCTIVITY, AS REPORTED BY RESPONDENTS

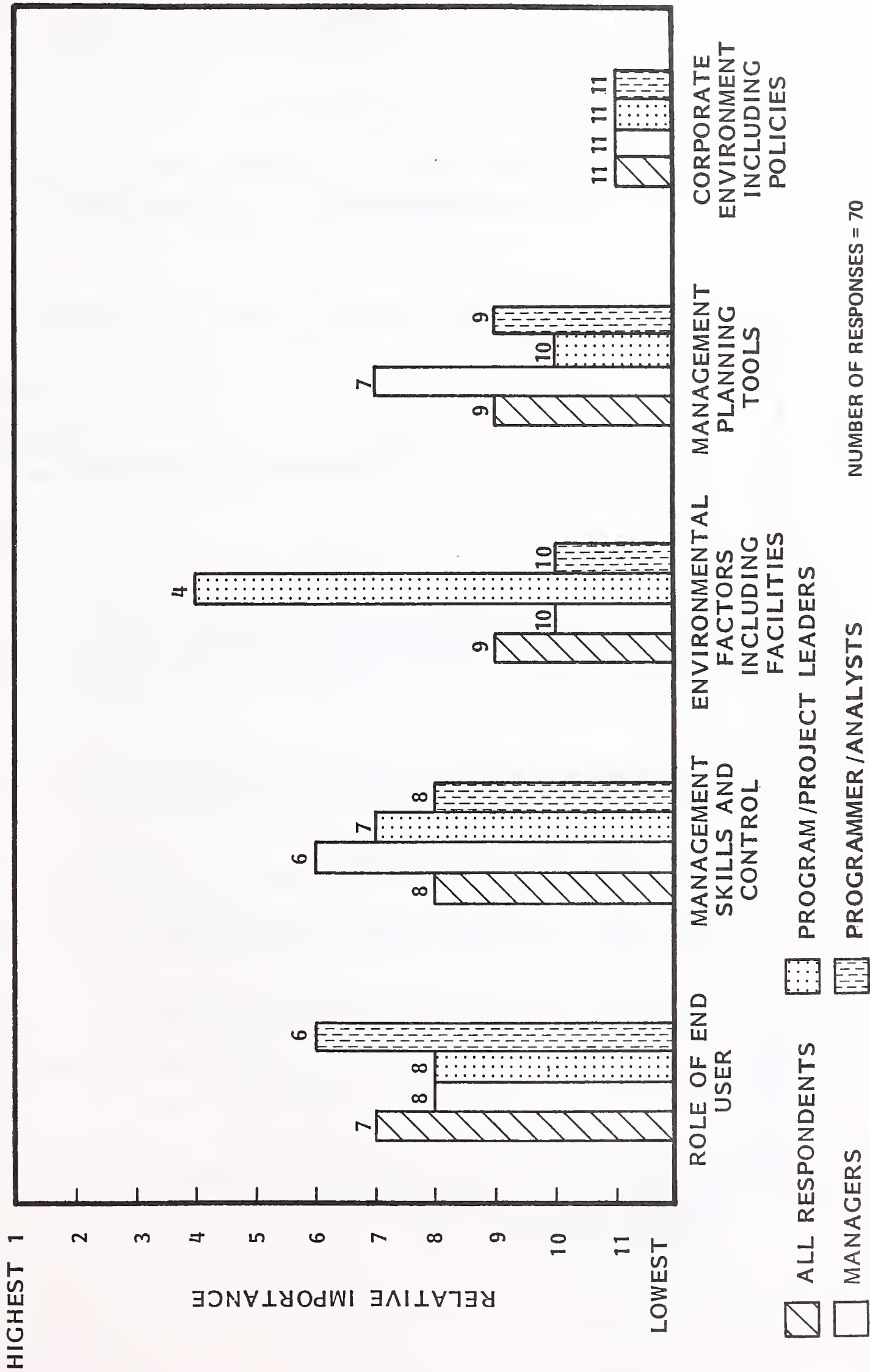


EXHIBIT III-12

ACTION TAKEN TO IMPROVE PRODUCTIVITY, AS REPORTED BY EDP MANAGERS

ACTION TAKEN	PERCENT* OF RESPONDENTS
INSTITUTE BETTER PLANNING	24%
PROVIDE PROGRAMMERS WITH ON-LINE DEVELOPMENT (CRTs + INTERACTIVE PROCESSING)	24
INCREASE STAFF EDUCATION AND TRAINING	22
GREATER USER INVOLVEMENT	14
IMPLEMENT SOFTWARE PRODUCTIVITY TOOLS	13
IMPLEMENT STRUCTURED DESIGN APPROACH	10

*MULTIPLE ANSWERS POSSIBLE
SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 136

- Three involved better tools.
 - The other three included better planning, training and user involvement.
 - Even the most common action (improved planning) was undertaken by less than a quarter of the respondents.
- These tentative initiatives are consistent with the earlier noted lack of:
 - A company productivity plan.
 - Localized productivity responsibility.

6. SUMMARY

- A picture emerges, then, of a DP function that is very tactically focused on tools and specific practices, and does not take a global or strategic view. It is saying, in essence:
 - "If only we had people who were a little better, with better tools, we could do a good job and make people happy."
 - "We don't have to look outside of our department and we don't have to rethink the larger issues of where we fit into the organization and how we should manage."
- This does not bode well for EDP managers, since this picture comes close to that drawn by their harshest critics, who consider them insular, inward-looking, narrow technicians. This profile is a recipe for:
 - A "more-of-the-same" mentality, which will have only marginal effects on existing problems.

- A data processing function that cannot serve the interests of the overall organization, since it is ignoring larger issues.
- On a more personal level, EDP managers cannot lead satisfying and successful careers in this environment. An EDP manager in this position will be condemned to a self-made fate, both master and victim of doubletalk.

7. A PRELIMINARY DIAGNOSIS AND DEFINITION

- The core of the problem relates to the vague definition of "productivity" in a data processing setting.
- In many settings, productivity is precisely a matter of improving the relationship between volume and cost of production.
 - The quality standards of the output are normally given: steel costs per ton implicitly assume that steel of at least the same grade will continue to be produced.
- In data processing, quality standards are still in the process of being devised. Certainly they are not yet universally accepted.
 - There are no fixed points of reference in deciding between the conflicting demands of output volume, costs and quality.
 - Some writers and practitioners of productivity seem to assume that, since quality is not defined, it does not matter. Therefore they focus only on the volume and cost factors.
- An illustration from the history of engineering may be helpful. During the mid-19th century there was a railroad-building boom that was comparable in many ways to the current impact of EDP. Railroads needed thousands of

bridges, for which there were many competing theories and designs. Cost and the time needed to erect the bridge were selling points for competing systems, both in-house and vendor-supplied.

- Unfortunately, the bridge builders had not fully mastered the design problems associated with working with new materials under heavier and heavier loads. In the 1870s alone, nearly 5,000 bridges collapsed, many involving loss of life.
- It was not until bridge engineering was able to specify in advance what the critical characteristics of a good bridge were that questions of cost and erection time became of real importance.
- If a similar diagnosis is applicable today, then EDP must reintroduce quality as the primary productivity dimension and motivator.
- Production of quality systems will not only make DP credible, but will also produce long-lasting results in other areas as well.
- Top management will have increased confidence in data processing. This confidence will encourage:
 - Listening to DP management on non-DP issues.
 - Expanding the breadth and depth of DP throughout the corporation.
- Users will see DP as a solution to their problems, rather than the cause or source.
 - Users will be more cooperative.
 - They will be more likely to take part in systems development, rather than having to be ordered or persuaded to participate.

- DP staff morale will increase, and turnover should decrease, since:
 - . They can take pride in the better product they will be building.
 - . They will no longer have to be defensive when dealing with users.
- Resources devoted to maintenance will be reduced since:
 - . Sturdier, more reliable systems will be built.
 - . Part of a quality strategy is to build systems that are easier to maintain.
- The importance of quality systems will be detailed explicitly in later sections of this study, and will be implicit in all that follows.

C. PRODUCTIVITY THROUGH EDP USER EFFECTIVENESS

I. THE FORGOTTEN USER

- In Section III. A. 1 the lack of end user involvement was cited as a relatively serious problem by EDP management. However, the end user role was not seen as an issue of major importance in improving productivity (III. B. 4), nor was greater user involvement being pursued by more than one out of seven EDP departments (III. B. 5).
- This tension, or inconsistency, is not unusual when dealing with or discussing the role of the end user.
 - In theory, the end user is the beneficiary of computers and data processing.

- In practice, vendors and data processing personnel have certainly reaped most of the rewards of computers.
- It is common for EDP personnel to feel that they know the users' operation better than the users do. But there is always a collective user memory of exceptions and special requirements that users are unable or unwilling to share with the DP know-it-alls who often don't have the patience to sit through a typical user's rambling discourse.
- Consequently, it should not be surprising that the EDP department sees the requirements definition process as its most critical inhibiting factor to improving productivity (III. B. 2).
 - However, as the means of overcoming this weakness, they look to better DP personnel and tools.
 - When users are finally permitted to inspect "their" systems after completion, out comes the age-old response: "It's a very nice system, but what I really wanted was ..."
 - Whatever time was allegedly saved by not bothering to talk to end users is then lost many times over in revamping the system to meet the users' true needs - a tremendous drain on productivity.

2. THE LARGEST SOFTWARE PRODUCTIVITY FAILURE

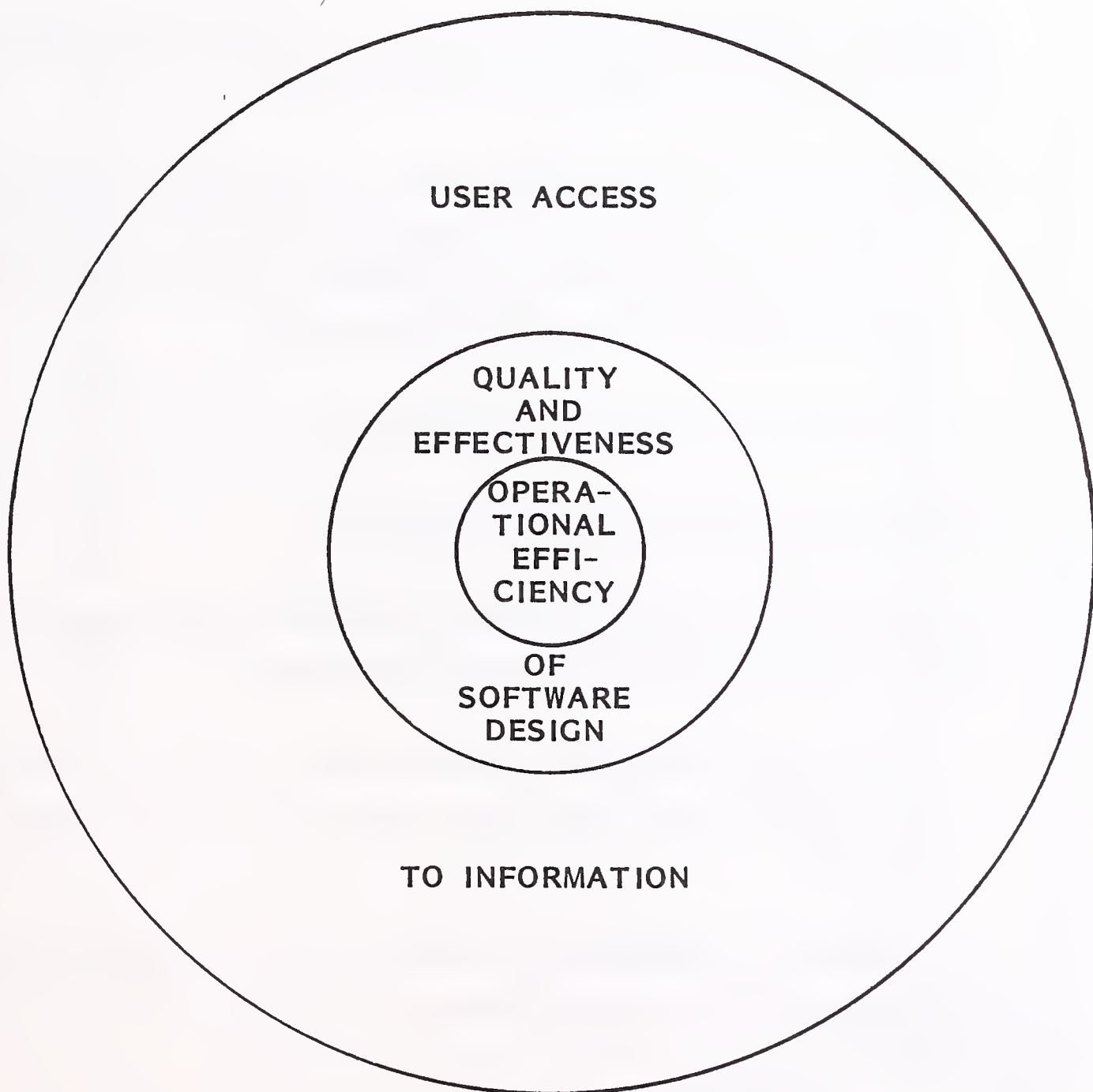
- In fact, it is precisely the failure of EDP to adequately meet the business information needs of the user that is the largest drain on potential corporate productivity improvement.
 - No marginal improvement in the development rate of software can begin to compare with the corporate benefits gained through increased marketing power, better forecasting of revenues and expenses, more comprehensive control of production facilities, improved automation of

engineering design, or any other corporate business function that requires better, faster, more accurate, more comprehensive or easier-to-use information.

- In this respect, EDP or MIS productivity can be imagined as a series of concentric circles, in which the innermost, smallest, and easiest to achieve is improved EDP operating efficiency, as shown in Exhibit III-13.
 - This aspect is usually achieved by installing larger hardware with better cost/performance ratios.
 - Tuning of software and better operating procedures are also effective at this level.
- Moving outward, the next level of EDP productivity must come from higher-quality software; i.e., software that is more reliable, more easily maintained, more extensive, and so forth.
 - Improvements at this level come from more effective control, design and testing methods; use of data base technology; greater integration of the software development process; and better access to software development, maintenance and documentation resources.
- The outside level of software productivity is the most comprehensive, but also the most difficult to achieve. It consists of facilitating the end users' access to, and control of, the business information they need for survival, growth and profitability.
 - Improvements at this level appear to be a matter of degree of user access, but often turn out to be a difference in the kind of user application.
 - Particularly productive at this level is the user's own ability (usually lacking at lower levels) to experiment with information in a variety of

EXHIBIT III-13

RELATIVE SCOPE OF EDP PRODUCTIVITY TARGETS



"what-if" formats, and to furnish the possibility for innovation in a less risky context than committing thousands of dollars to a real-life experiment.

- Implicit at this level are a comprehensive data base and an easy-to-use retrieval system, plus the degree of user understanding of, and confidence in, the EDP function that encourages direct user interaction with both data and process.

- Thus the two innermost levels of productivity must be in place before much of the outermost level of productivity can be achieved.

- Attempting to bypass these stages not only raises the risk of failing to achieve the outermost level of productivity, but increases the penalty for failure by destroying the credibility of the EDP organization, and squandering the resources of the corporation.

- At this point, reorganization is essential

3. THE CRITICAL NEEDS DEFINITION PROCESS

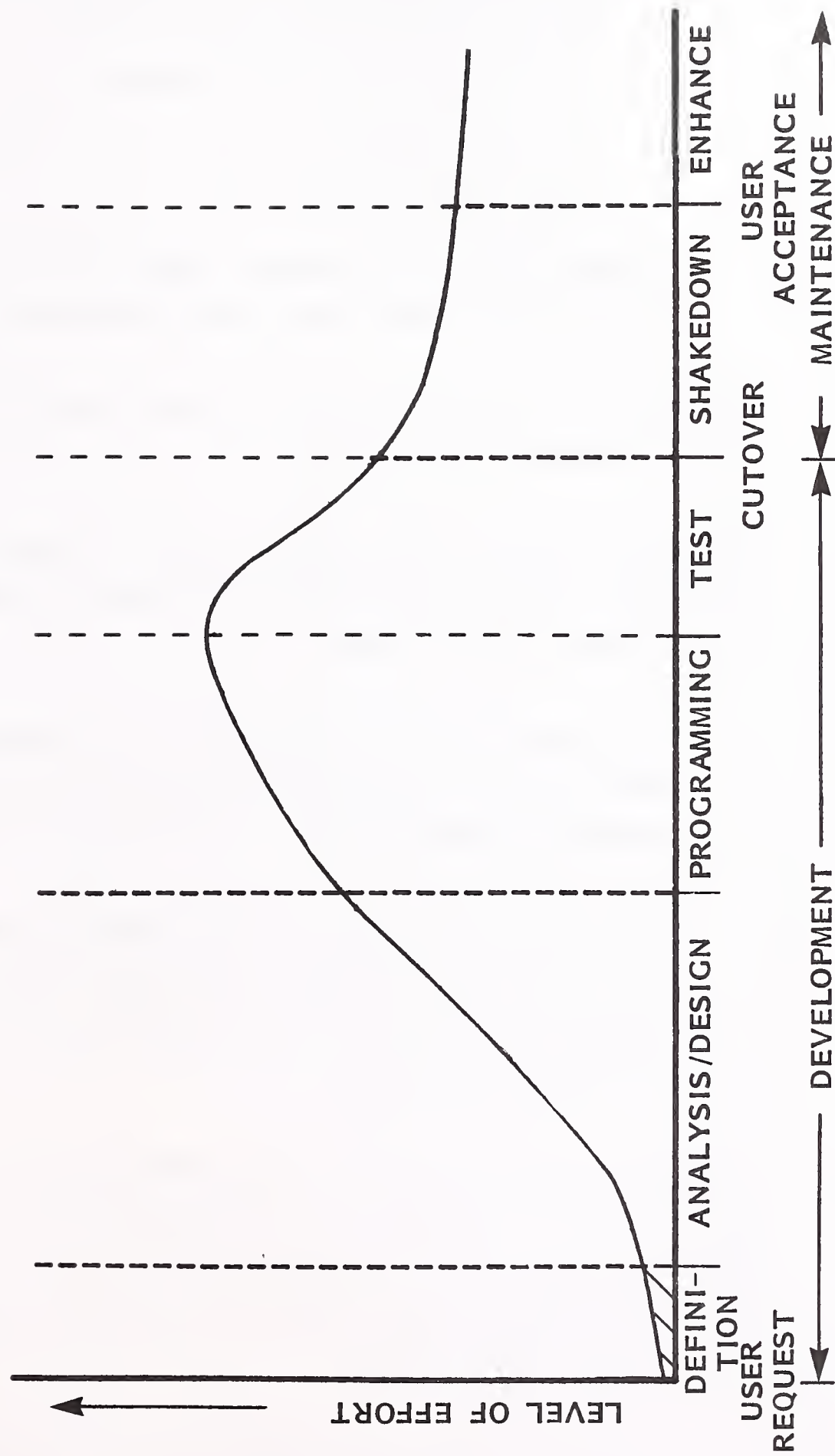
- Currently, the needs definition process consumes the least amount of time and effort on a typical project, as shown in Exhibit III-14.

- This brevity results partly from the desire on the parts of EDP and user management alike to see some results (i.e., code and output, respectively).

- However, a substantial part of the reason is that the requirements definition/system design portion of the project is:

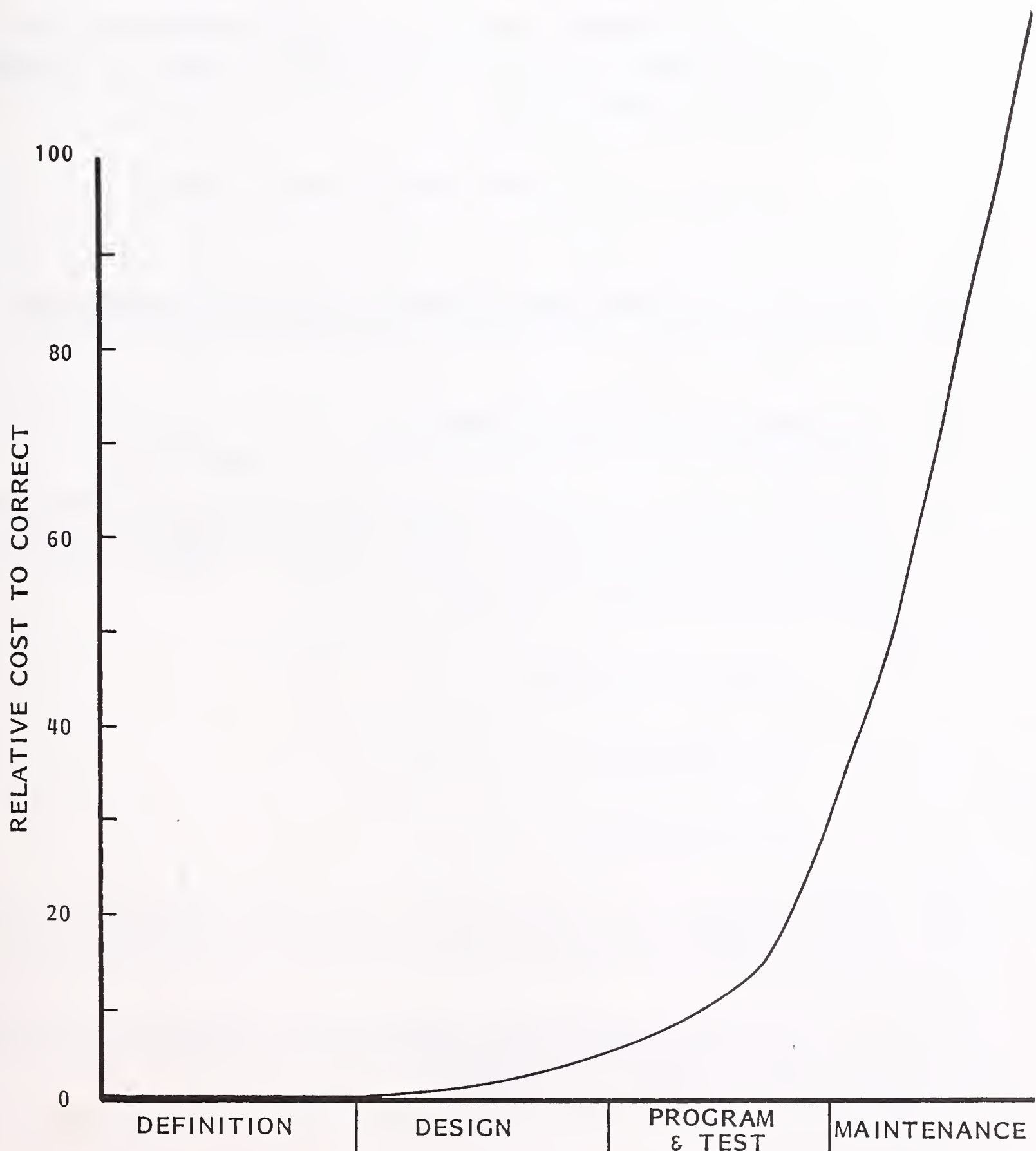
- . Most difficult.

TYPICAL PROJECT WORKLOADING CURVE



- Least understood by outsiders. ("What do you mean, what kind of billing system?")
 - Most impervious to mechanization.
 - Least like data processing.
- Consequently, most computer people want to get on to the warm, comfortable technical tasks as soon as possible.
- This rush to programming is dangerous, given the risks inherent in leaving the definition/design phase too early.
- After definition and early design is past, the cost to correct errors and omissions mounts at a terrifying rate, as shown in Exhibit III-15. When errors are discovered at a later stage:
 - Their correction throws the cost and time estimates off schedule. This single factor encompasses the top four productivity problems of EDP management (see III. B. 1).
 - The temptation is high not to fully correct these problems in order to save face as well as time and expense. This results in an inferior system that will have to be corrected later, at an even higher cost, or "enhanced" prematurely.
- The problem is not resolved by simply allocating some number of additional resources for the requirements definition phase the next time around, or, in a more sophisticated and potentially disastrous step, by arbitrarily instituting one of the development approaches that front-loads much of the development effort to the pre-coding phase.
 - If the relationship and expectations of the user and EDP are not clarified (i.e., changed), sending five analysts, rather than one, into a

EXHIBIT III-15
IMPACT OF CORRECTING ERRORS DURING SOFTWARE
APPLICATION SYSTEM LIFE CYCLE



user's office for the first (and only) needs-assessment interview will not have any positive effect on the project.

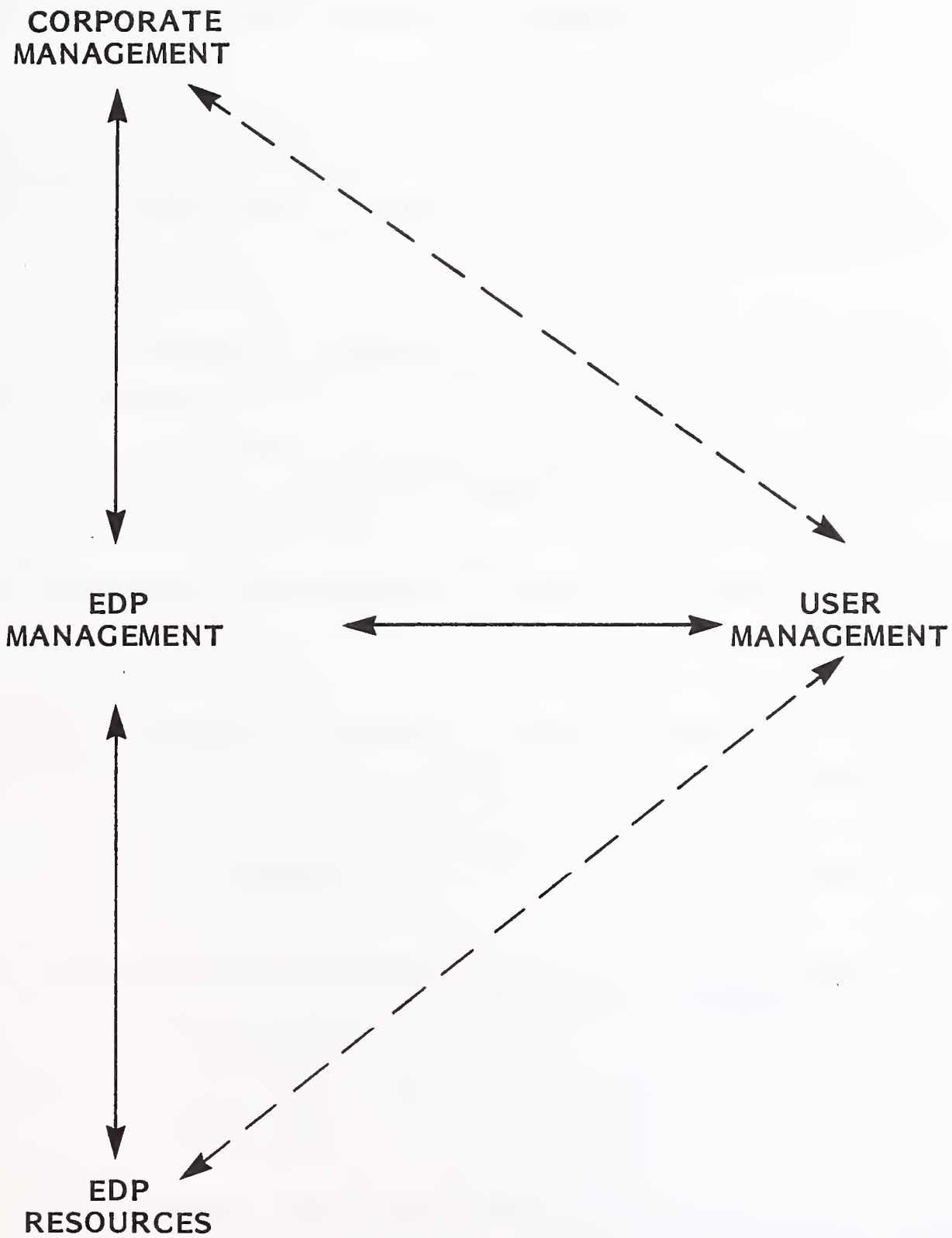
- To make the needs definition process really work, systems people have to "get inside the user's head" and think like a user.
- A final, sobering thought: providing each analyst and programmer with the finest set of productivity tools in the world will not change any individual's wrong thinking or faulty attitude.
- At its base, the productivity problem is caused by people.

D. PRODUCTIVITY IMPROVEMENT THROUGH EFFECTIVE MANAGEMENT

I. MANAGEMENT INTERRELATIONSHIPS

- Focusing on users is a necessary precondition to productivity improvement. However, it is not sufficient. To make it work, management must make a broad-based commitment at all levels.
 - Corporate management.
 - EDP management.
 - User management.
- EDP management can be seen as either "in the middle" or in a position to take control, depending largely upon its own initiatives, as shown in Exhibit III-16.
 - Corporate management will normally look to EDP management for data processing plans and strategies.

EXHIBIT III-16
THE MANAGEMENT NEXUS



◄ — — ► PREFERRED RELATIONSHIP
◄ - - - ► UNDESIRABLE RELATIONSHIP

- Similarly, user management would, all things being equal, view EDP management as its source of service and advice on data processing matters.
- Data processing resources would normally be under the control of EDP management.
- This situation can change radically, with user management opening other channels of communication and control, if it feels that EDP management is preventing user needs from being met.
- However, it is essential that all management levels shoulder the burden and stand firmly and publicly in support of productivity improvement as a goal.

2. EDP MANAGEMENT'S EXTERNAL ROLE

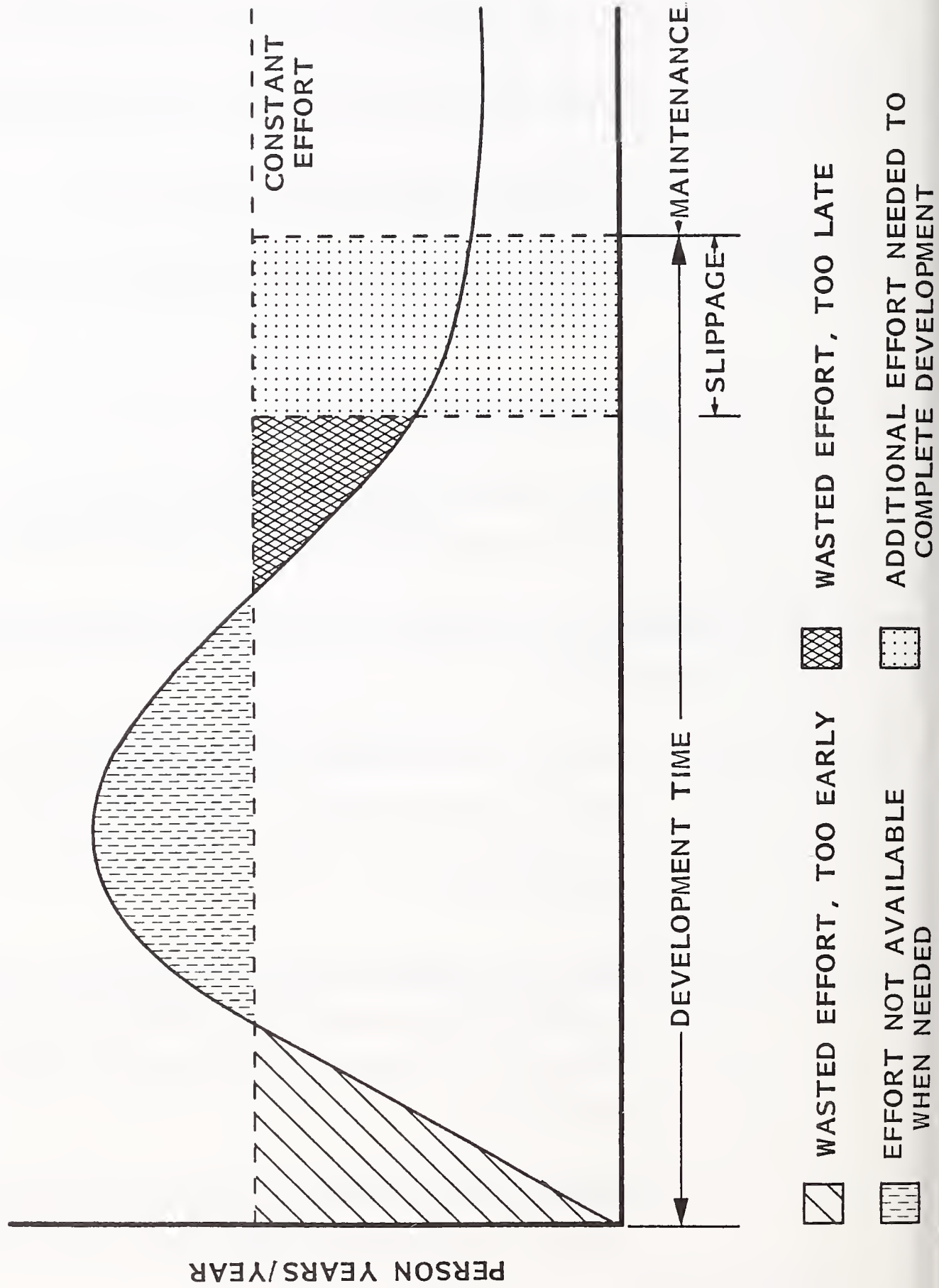
- EDP management now views its role as essentially inward-looking (as discussed in III. B. 4).
- If EDP management is going to succeed, it must turn its view outward and upward, and:
 - Seek out users to understand their problems.
 - Set up progressively more comprehensive organizational structures to solidify the user-EDP links.
- Similarly, EDP management must:
 - Set up adequate EDP planning and control mechanisms.
 - Integrate them with corporate-wide planning and control structures.

3. EDP MANAGEMENT'S INTERNAL ROLE

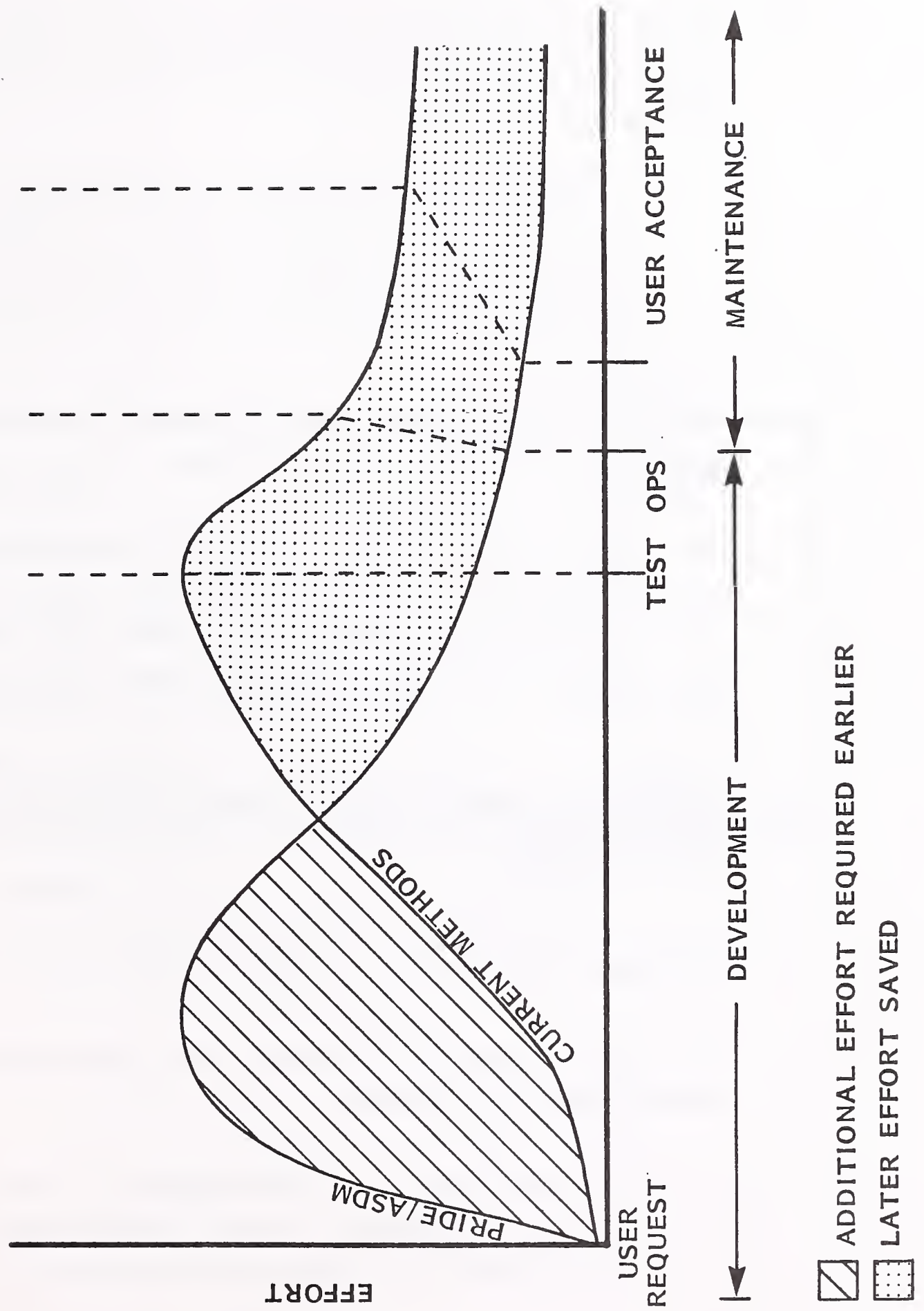
- Internally, EDP responsibility for improving productivity extends to:
 - Personnel recruitment, training, career development and motivation.
 - Providing an adequate working environment.
 - Assuring dependable, adequate access to computer development and testing resources.
 - Fostering technical growth.
 - Establishing satisfactory internal planning and control mechanisms, including keeping the channels of communication open.
- The effective management of personnel resources can have a critical impact on productivity.
 - Projects must be staffed and managed to take account of the development life cycle, as shown in Exhibit III-17. Poor timing of additions to the project team can easily swing the cost and completion date of a project by 10%.
 - One of the effects of choosing an appropriate life-cycle approach, such as PRIDE/ASDM, is that it can minimize longer-term resource requirements, even though it increases front-end requirements, as shown in Exhibit III-18.
 - However, as discussed in III.D.2, such front-loading can only be beneficial in a receptive organizational environment.
 - Furthermore, life-cycle savings will usually not be achieved after the second project, due to the lengthy learning curve involved.

EXHIBIT III-17

THE NEED FOR RESOURCE MANAGEMENT OF SOFTWARE DEVELOPMENT PROJECTS



THE IMPACT OF DESIGN METHODOLOGIES ON THE LIFE CYCLE



4. MANAGEMENT BALANCE

- Management must balance its internal and external roles. To neglect one at the expense of the other runs the risk of negating much of the productivity achievement.
 - In a case study for this project, an EDP department performed very well in all areas except user relations, as shown in Exhibit III-19. Unfortunately, this one defect undermined many of its accomplishments elsewhere.
- A similar problem afflicts those installations that focus a large amount of attention on one segment of the development cycle.
 - Most of the better-known development tools focus on the coding phase of a project, as shown in Exhibit III-20. Since coding comprises less than 10% of total project effort, even a 50% improvement in coding would have less than a 5% overall impact on the typical project.

E. PRODUCTIVITY IMPROVEMENTS THROUGH EDP PERSONNEL EFFECTIVENESS

I. PERSONNEL UTILIZATION - A KEY FACTOR

- The single most important factor under the control of EDP management is the effective use of EDP personnel.
 - It is well known that individuals vary widely in their ability to understand requirements, produce adequate designs, write tight code, write understandable code, communicate with users, communicate with

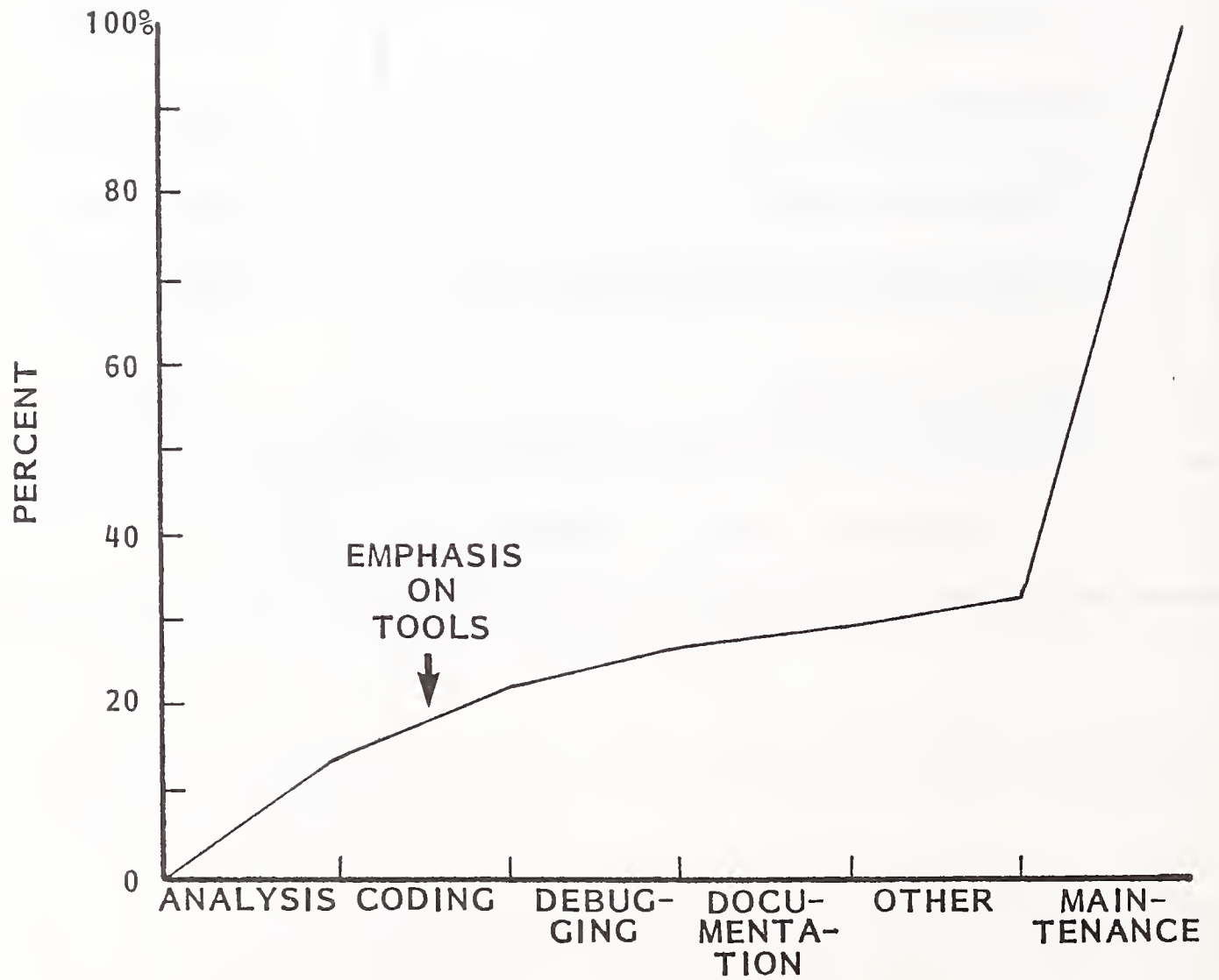
EXHIBIT III-19

CASE STUDY: THE XYZ ORGANIZATION

AREA	ACHIEVEMENT
● PERSONNEL	VERY GOOD
● ENVIRONMENT	VERY GOOD
● TOOLS AND AIDS	VERY GOOD
● MANAGEMENT AND ORGANIZATION	VERY GOOD
● USER RELATIONS	FAIR
RESULT = STILL A "PERCEIVED PROBLEM"	

EXHIBIT III-20

RELATIVE COSTS OVER SYSTEM LIFE



each other, find errors and do almost every other job required of the programmer or analyst.

- Exhibit III-21 shows ranges of individual programmers' performances that have been quoted in various INPUT studies. Any manager can add similar experiences to the list.
- Even wider programmer ranges than these are evident among systems analysts, where the difference on the high side extends to infinity in the case of individuals who cannot cope intellectually with the complexity of the system to be analyzed.
 - Many organizations now complain about the combination of functions incorporated in a programmer/analyst title, citing the resultant lowered competency within the systems analysis function.
 - Although the two functions are related, they require a different conceptual approach to problem solving:
 - Analysts must continually look for constraints upon the design, and can never be sure that all constraints have been properly specified.
 - Programmers must work within defined constraints, and should always know when all specifications have been met.
- The reduced competency cited above resides not so much in systems analysis as in business analysis.
 - The apparent shortage of programmers will not, in fact, be as severe in its effects as the largely unheralded shortage of people who understand business.

EXHIBIT III-21

RANGES OF PRODUCTIVITY

•	DEBUGGING	26:1
•	CODING	25:1
•	EXECUTION SPEED	13:1
•	DEVELOPMENT/CPU USAGE	11:1
•	DEFECT REMOVAL	10:1
•	LINES OF CODE	5:1

- In the future, simple programming applications will be handled by standardized software packages and software modules, perhaps contained within silicon chips.
 - Complex and/or unique software applications will require much higher skill levels in personnel, who will be subject to the same kinds of demand rates that systems programmers and data communication specialists already experience.
- It is critical that EDP management find ways to capitalize on individual strengths already existing within the organization.
 - For example, the "Chief Programmer Team" concept assigns the most technically complex or critical aspects of a project to the most technically adept individuals, without regard to their organizational position.
 - An alternate approach is to establish a technical services group to concentrate on identifying, isolating and finding ways to work around the software aspects that are most troublesome to EDP personnel of average competency.
- This is perhaps the most difficult of the manager's responsibilities. ("Manager" here means each member in the management chain, from first-line supervisor to top person in the hierarchy.)
 - It should be clear that managerial skills do not come automatically with the promotion to managerial ranks, while high technical competency is also a scarce resource. Both must be developed through training and experience, and some individuals are never going to be expert, or even comfortable, exercising either of them.

- Few things have been as damaging to data processing as the assumed programmer-analyst-manager path, which makes as little sense as a bricklayer-architect-real estate developer progression.
- The negative impact on productivity of poor managerial skills and practices goes far beyond the negative impact any individual programmer or analyst can cause, because the effects are magnified by the number of people reporting to, or coming in contact with, the poor manager.
- Thus management motivation was ranked second only to personnel motivation as the area of most importance in achieving greater software productivity (as discussed in III. B. 4).
- A partial solution is to provide a financially attractive, technical career path that does not require managerial responsibility for advancement, thus retaining the technical expertise where it does the most good, and also leaving the management role free for people with managerial talent.

2. PERSONNEL SHORTAGE

- During 1980, EDP managers have frequently discussed the size of the personnel shortage.
- Underlying this issue is a hodgepodge of misconceptions, uninformed estimates, guesstimates, and managerial reactions bordering on mass hysteria.
- To understand the role of the current personnel shortage in the overall issue of productivity in software development, it is necessary to examine the factors used to forecast supply and demand for EDP professionals.
- There is no standard base year for measuring or estimating changes in EDP employment, nor are there any standard methodologies for producing personnel forecasts. Most published figures rely in one way or another upon government-

generated statistics and straight-line forecasts of employment, which are all related in some way to estimates of computer hardware installed.

- Under these circumstances, gross errors in estimating may arise from a failure to define terms. An Apple computer may be weighted equally with an IBM 3033 when counting the number of computers installed, for example.
- Seldom are allowances made for double-counting people who may function primarily as programmers or analysts, while their actual job title is EDP manager, bookkeeper or consultant.
- Attempts to reconcile forecasts in this area are fraught with difficulties and frustration, since independent estimates from different parts of the same organization may each be valid, depending upon the assumptions on which they are based. Unfortunately, these assumptions are rarely made public.
- The Bureau of Labor Statistics in the U.S. Department of Labor (DOL/BLS) estimates that in 1976 there were 230,000 programmers and 160,000 systems analysts. The BLS expects these occupations to increase at average annual growth rates of slightly over 2.5%, reaching 285,000 and 200,000 respectively in 1985.
 - In addition, the BLS expects 0.5% to 1% of job openings to occur annually in these categories, due to replacement of current personnel.
- Growth plus replacement would thus provide a total of approximately 8,900 programmer openings and 5,300 systems analyst openings in 1980, according to one set of Department of Labor estimates (assuming a linear trend from 1976). However, another set of DOL/BLS estimates puts these annual needs at 23,000 and 27,000 respectively.
- The major sources of supply for entry-level positions in EDP are general college graduates, computer science or data processing majors at the

bachelor's and master's degree levels, and graduates of two-year programs in data processing from recognized academic institutions.

- Diploma holders from vocational education programs of various sorts have generally not been well accepted into the business community except at the level of operator or data entry personnel, and these institutions are not considered a promising source of programmers and analysts in the near future. Nevertheless, the number of vocational DP graduates is substantial, exceeding 20,000 annually according to the U.S. Department of Education.
- Current projections of academic graduates in computer science, data processing and related fields have been tracked for more than a decade by Professor John Hamblen of the University of Missouri-Rolla. His most recent projection also covers the number of degrees awarded in 1976-1977, as shown in Exhibit III-22.
- According to Professor Hamblen, the need for computer professionals is growing at approximately 10% per year, and currently stands at approximately 116,000 per year, including computer operators, managers, and other DP personnel such as tape librarians.
 - These figures slightly exceed the higher of the two DOL/BLS estimates of demand, and translate to a need for 55,000 programmers and analysts per year.
- The only category that appears to be producing candidates at an acceptable rate is the two-year (diploma) level, if both academic and vocational graduates are counted.
 - However, as noted earlier, these graduates are most frequently employed below the programmer or analyst level.

EXHIBIT III-22

REPORTED ENROLLED MAJORS AND DEGREES AWARDED IN COMPUTER SCIENCE, DATA PROCESSING AND RELATED FIELDS: 1976-1977

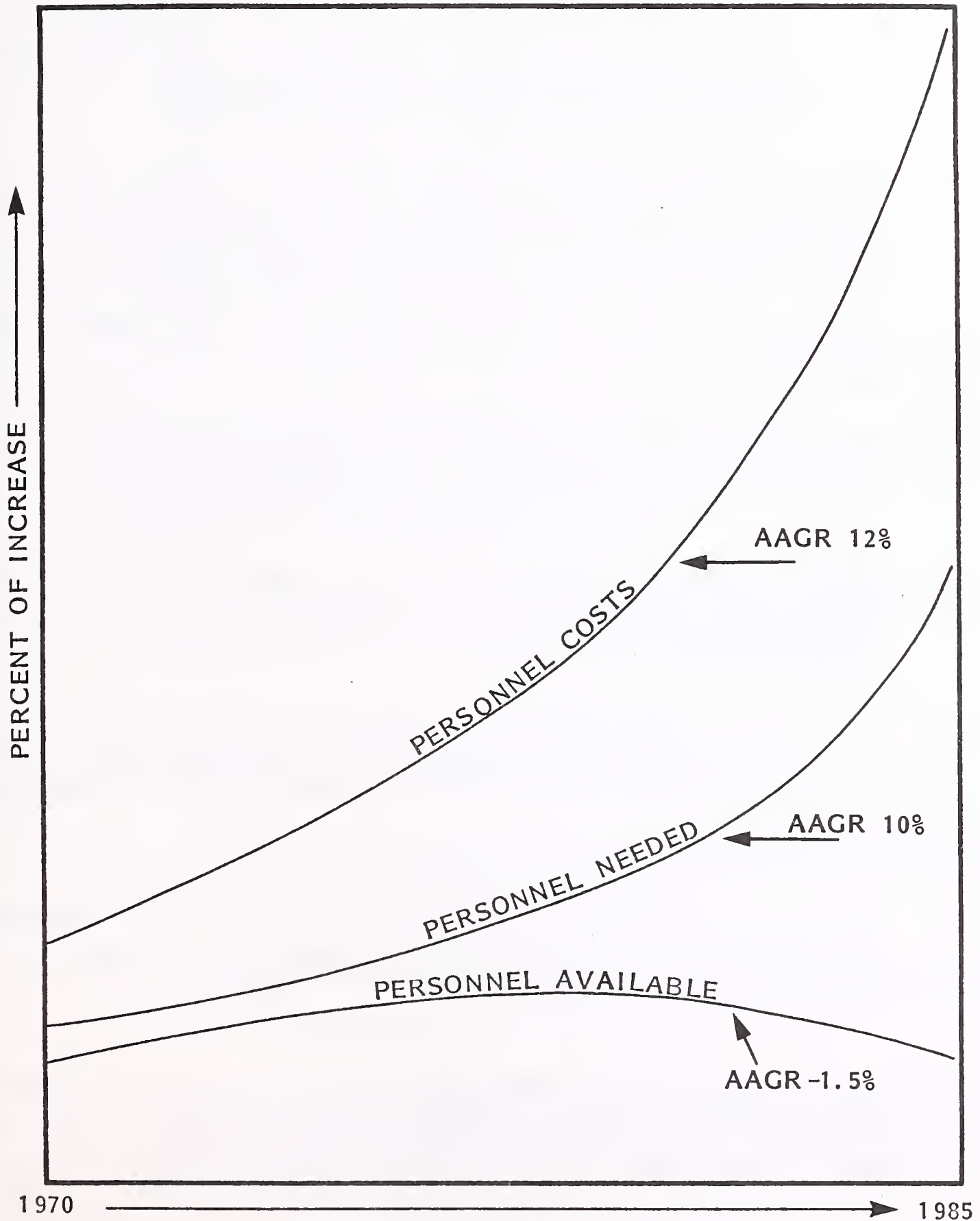
CATEGORY	REPORTED TOTAL	ESTIMATED TOTAL POPULATION*
ENROLLED MAJORS, UNDERGRADUATE	61,062	105,190
GRADUATES, ASSOCIATE LEVEL	4,801	8,271
GRADUATES, BACHELOR'S LEVEL	5,520	9,509
GRADUATES, MASTER'S LEVEL	1,835	3,161
GRADUATES, DOCTORAL LEVEL	200	345
GRADUATES, TOTAL ACADEMIC	12,356	21,286
GRADUATES ENTERING LABOR POOL FROM VOCATIONAL PROGRAMS	N/A	19,041
GRAND TOTAL GRADUATES AVAILABLE	-	40,327

* REPORTED TOTALS x NUMBER OF INSTITUTIONS IN POPULATION/NUMBER OF INSTITUTIONS REPORTING

- Bachelor's level graduates are only sufficient to satisfy one-sixth of the estimated current demand, although they doubled in number from 1974 to 1977.
- Master's level graduates are only meeting one-eleventh of the estimated current demand (i.e., at supervisory levels and above), and increased in number by only 10% from 1974 to 1977.
- None of these figures takes into account the number of people who enter the data processing field possessing a college degree in fields other than data processing or computer science.
 - Unfortunately, no estimate exists of the number of people in this category.
 - However, the "baby boom" of the fifties has already graduated from college, and the number of 18- to 24-year-olds in 1985 will be 27,853,000, or almost 4% lower than in 1978.
- Thus Exhibit III-23 begins to take on new meaning if the demand for computer professionals continues to grow at 10% per year, whether the supply grows slightly, remains constant or actually shrinks by up to 1.5% a year.
- INPUT's survey of large-scale organizations (greater than \$1 billion in annual revenues) indicates a more modest growth plan in the number of EDP employees, ranging from 0-10% annual growth, but tending toward the lower end of the range.
 - Several organizations have made a conscious attempt to place a size limit on their EDP organizations, and have in fact been able either to hold their head-count stable or to reduce it over the last five to ten years.

EXHIBIT III-23

ANTICIPATED PERSONNEL COSTS AND SHORTAGE



- Concurrently, these organizations have placed a heavy managerial emphasis on continually upgrading the caliber of their staff, and upon controlling turnover within the organization by providing good working conditions, competitive (above-average) salaries, and state-of-the-art technologies.
- Justification for this strategy was a lower total expenditure and less disruption of the system development effort. However, no formal post-audits have been conducted to validate this strategy.
- While the average projected personnel growth rate for these organizations is between 0% and 10%, individual components within the organization may have already experienced, or may be planning to achieve, a 30-50% increase in staff within a single year (calculated on a small initial base, generally less than 30 people).
- This personnel shortage implies that the requirements of individual organizations are far more significant than overall industry averages.
- Larger organizations typically have greater resources to devote to planning, training and developing of staff from within the organization than smaller organizations do.
 - They can also more easily justify and integrate contract programmers within the EDP staff to handle overloads or specific technical requirements.
- By the same token, it is often less easy for the larger organization to efficiently control the quality and costs of outside labor, because of the large numbers of people involved.
- On the balance, larger organizations that act immediately to project their EDP labor needs over the next three to five years should be able to meet at least half those needs through internal recruiting and personnel development.

- These efforts must be coupled with an intensive campaign to retain experienced managerial and technical staff.
- The one situation that could unduly affect even the larger organizations would be their becoming engaged in competition with their own users when hiring programmers and/or analysts.
 - The trend toward greater use of personal computers and small business computers within large organizations is expected to accelerate over the near term, and could lead to competition in hiring practices as users discover that real programming expertise is required for any "personal computing" beyond the most routine of business applications.
- Medium-scale organizations will feel the greatest impact from the personnel shortage, both because they are typically in a steeper growth phase, and because they are typically having to cope for the first time with sophisticated applications requiring skills not resident in their existing EDP staff; i.e., primarily data base and telecommunications skills.
 - Competition will be most fierce for EDP personnel possessing these skills, as the demand rates soar by 20% or more per year.

3. STRATEGIES FOR DEALING WITH PERSONNEL SHORTAGES AND RECRUITMENT PROBLEMS

- The main thrust of all strategies is to avoid competing for those personnel categories that promise to be in shortest supply:
 - Experienced programmer/analysts (who not only may be overpriced, but of indifferent quality).
 - Computer Science graduates - Bachelor's degree and higher. (Many of these graduates would not be appropriate for a commercial DP setting in any event.)

- Reduce turnover: INPUT's survey has revealed that replacing departed staff accounts for two-thirds of the recruitment needs of a typical firm. This component represents the largest potential savings from a numbers standpoint, but is also the most difficult to deal with. (See the next section for a complete discussion.)
- Internal DP department upgrading: The same survey shows that internal trainees currently represent about one-third of all trainees. Upgrades present less of a rush and require a smaller investment than many externally acquired trainees. In addition, internal recruiting helps build organizational loyalty and cohesiveness.
- Intracompany transfers: Currently, the transfer rate of experienced staff into the DP department is well under 1%. At the same time there is usually a shortage of business analysts experienced in particular areas. Arrangements could be made with user departments to transfer analytically oriented staff to the DP area, either for a fixed or indefinite period. This staff would need moderate technical training and orientation to be effective. Use approved tests as a screening device, but utilize other judgmental factors as well.
- Disperse technical tasks to users: The investment for this strategy involves setting up:
 - "User-friendly" systems.
 - Associated software (e.g., query languages, report generators, etc.).
 - Terminals and printers.
 - Training materials and educators.
 - Backup and advisory staff.

- The ultimate goal is to enable users to perform much of their own analysis and programming, at least for routine maintenance and retrieval applications. Two pitfalls to avoid in the short term are:
 - Generating so much interest that more DP staff time is consumed.
 - Generating so much more, or so much inefficient, machine utilization that the ability to supply computer resources is affected.
- Employ graduates of two-year data processing programs.
 - Advantages are that they are fairly easy to get and already have training and some experience.
 - Disadvantages are that they require more training, and may not be ultimately promotable.
- Establish a "trainee-rich" organization. This strategy requires a different organizational approach that would maximize the use of trainees as opposed to more experienced staff. (See Appendix D for a complete description.)
- Don't be bound by traditional archetypes:
 - Choose quality and personal characteristics over experience.
 - Use non-salary incentives strategically, not arbitrarily, including:
 - . Flextime.
 - . Terminal at home.
 - Integrate contract programmers on project teams, not isolated assignments.

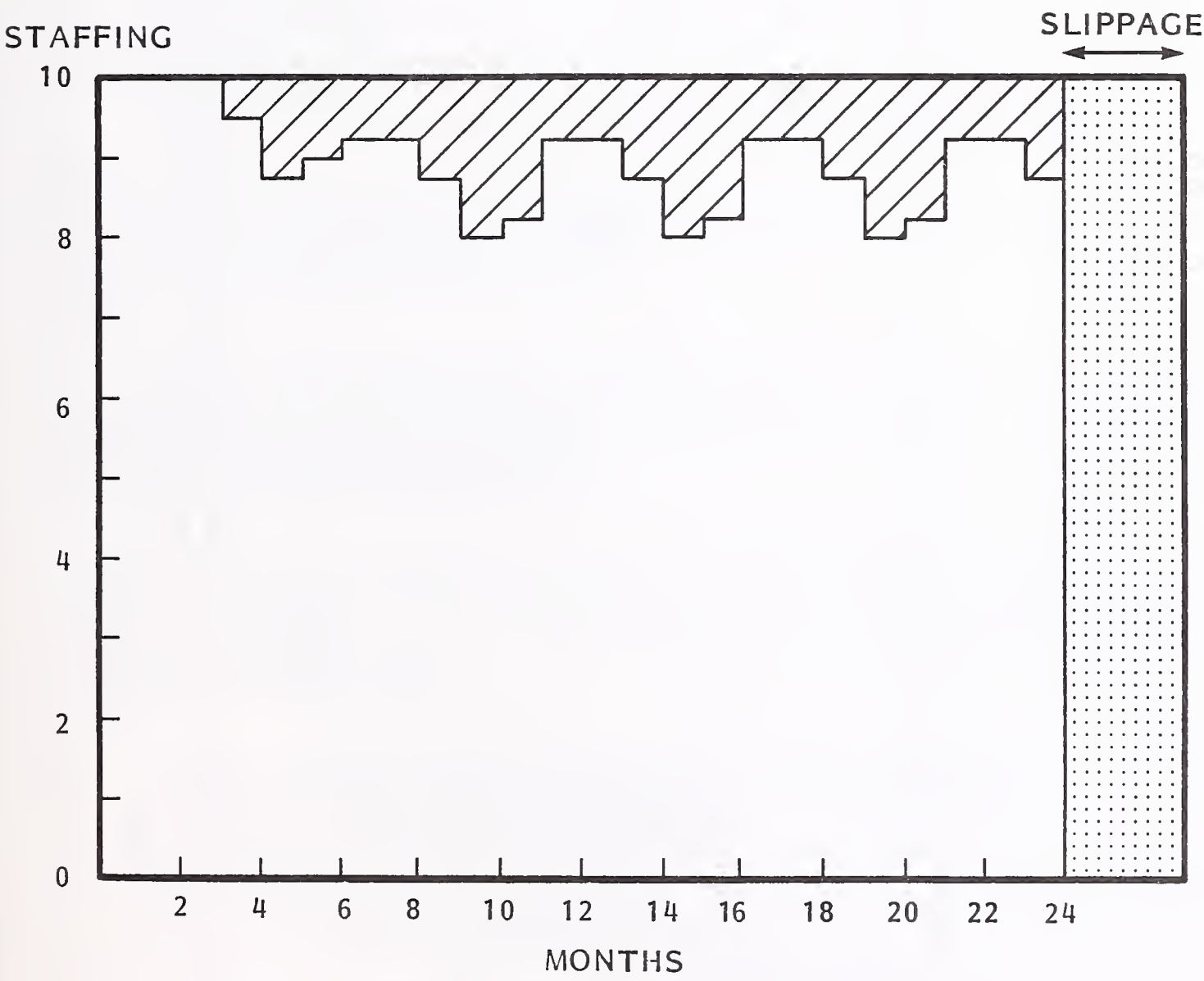
- Make use of part-time personnel who are familiar with your organization.

4. TURNOVER

- Turnover is an important issue, for both quantitative and qualitative reasons.
 - The effect of 20% turnover on a hypothetical two-year project is shown in Exhibit III-24: The project slips 10% even if no other problems or side effects occur.
 - Assumptions for this hypothetical case are:
 - Departing individuals are 50% effective during their last month before departure.
 - The project leader is 50% effective during the one-month recruiting period.
 - One month elapses before the new individual starts.
 - The new individual is 25% effective for the first two months, 50% effective for the next two months.
 - Departures are evenly spaced at five-month intervals throughout the project, beginning at the fourth month.
- INPUT's research indicates an average turnover rate among respondents of 12%, with an average addition rate of 18%, for a net increase of 6%, as shown in Exhibit III-25.
 - External losses account for five times the number of internal losses (to other departments).

EXHIBIT III-24

MINIMUM IMPACT OF 20% TURNOVER



TOTAL PROJECT = 240 PERSON MONTHS

MINIMUM IMPACT = 25 PERSON MONTHS

SLIPPAGE = 10.4%, OR 2.4 MONTHS



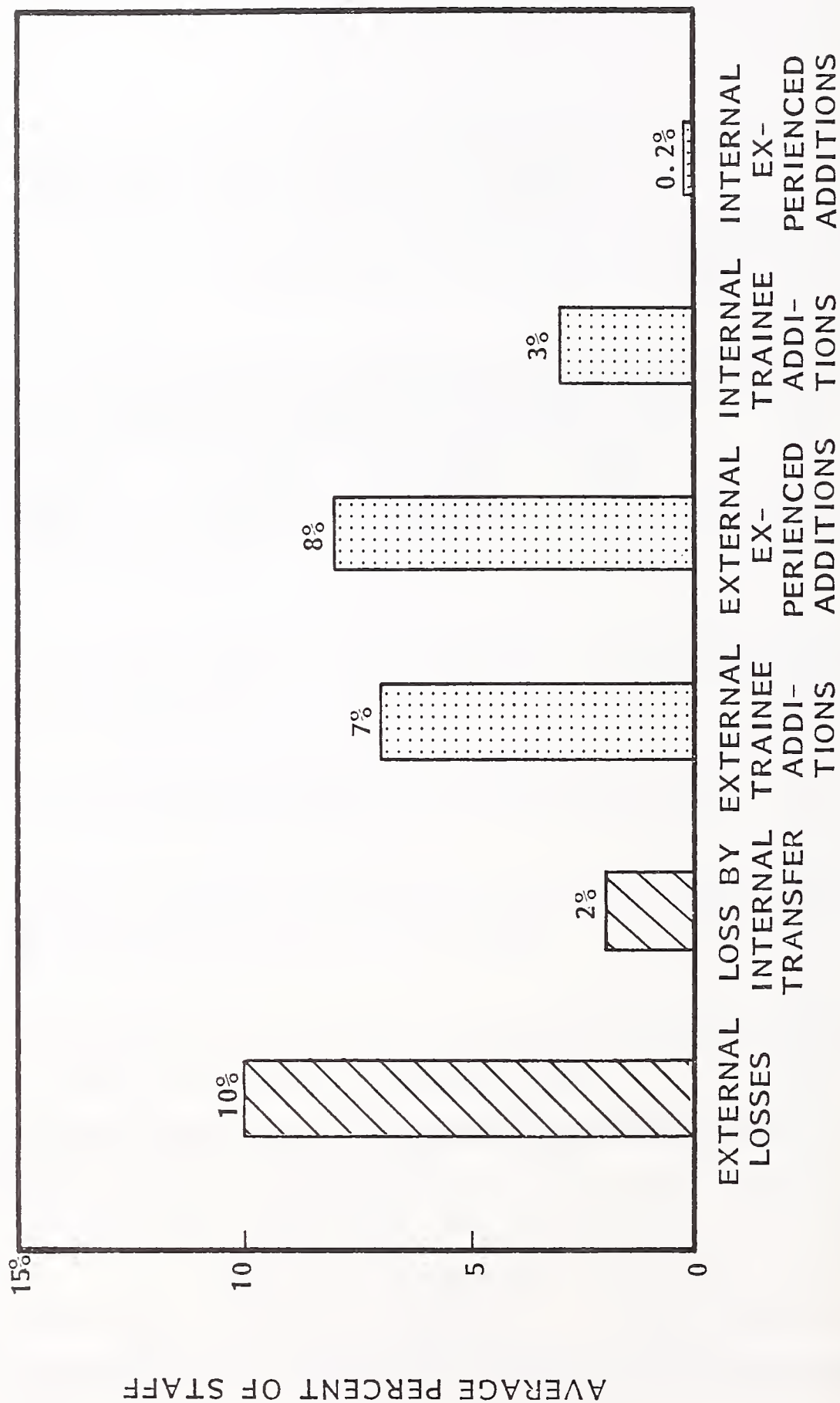
EFFORT LOST



ADDITIONAL EFFORT
REQUIRED

EXHIBIT III-25

DISTRIBUTION OF ANALYST/PROGRAMMER TURNOVER AS REPORTED BY EDP MANAGERS

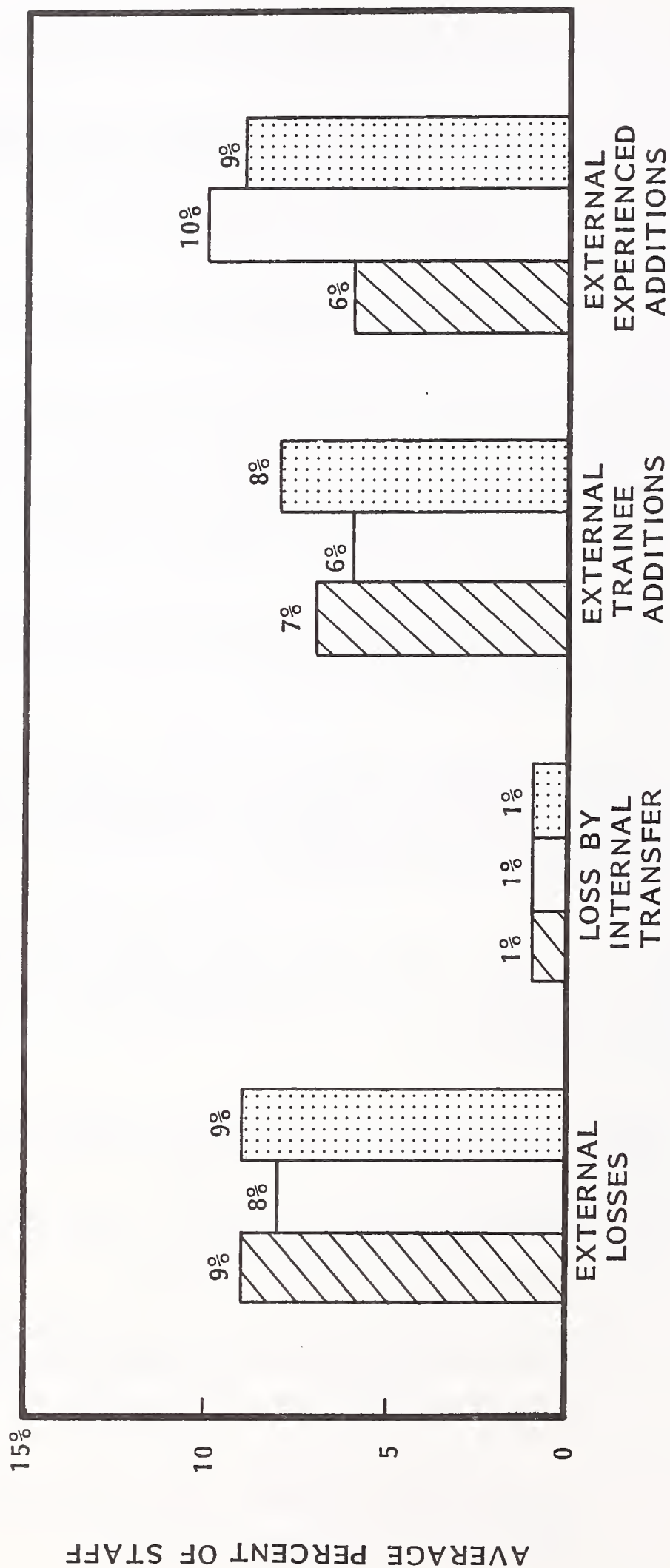





SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 300

- Trainees make up about half the new hires.
- Company size does not appear to be a significant factor, except that medium and small firms tend to hire more external, experienced additions, as shown in Exhibit III-26.
- This is the normal growth pattern for firms in these size ranges.
- Perceptions vary by staff level as to the extent of turnover. Managers appear to think their programmer/analyst turnover is less than other firms in the surrounding area, while programmer/analysts themselves think that their firm's turnover is significantly higher, as shown in Exhibit III-27.
 - Programmer/analysts also overstate their own firm's turnover by a factor of two.
- A very significant finding that INPUT made in the course of this study is that project turnover from internal EDP department transfers probably equals total turnover from all other sources.
 - In many cases, this turnover is even more disruptive than other types of turnover since it occurs suddenly due to an emergency elsewhere in the organization.
- There are several ways a department can deal with turnover:
 - Follow strategies to improve morale and motivation in general (as discussed in the following section).
 - Follow documentation and development strategies that minimize the importance of any particular individual.
 - Consider end-of-project or end-of-phase bonuses to prevent disruptive departures on key projects or phases.

EXHIBIT III-26

DISTRIBUTION OF ANALYST/PROGRAMMER TURNOVER AS REPORTED BY EDP MANAGERS BY COMPANY SIZE

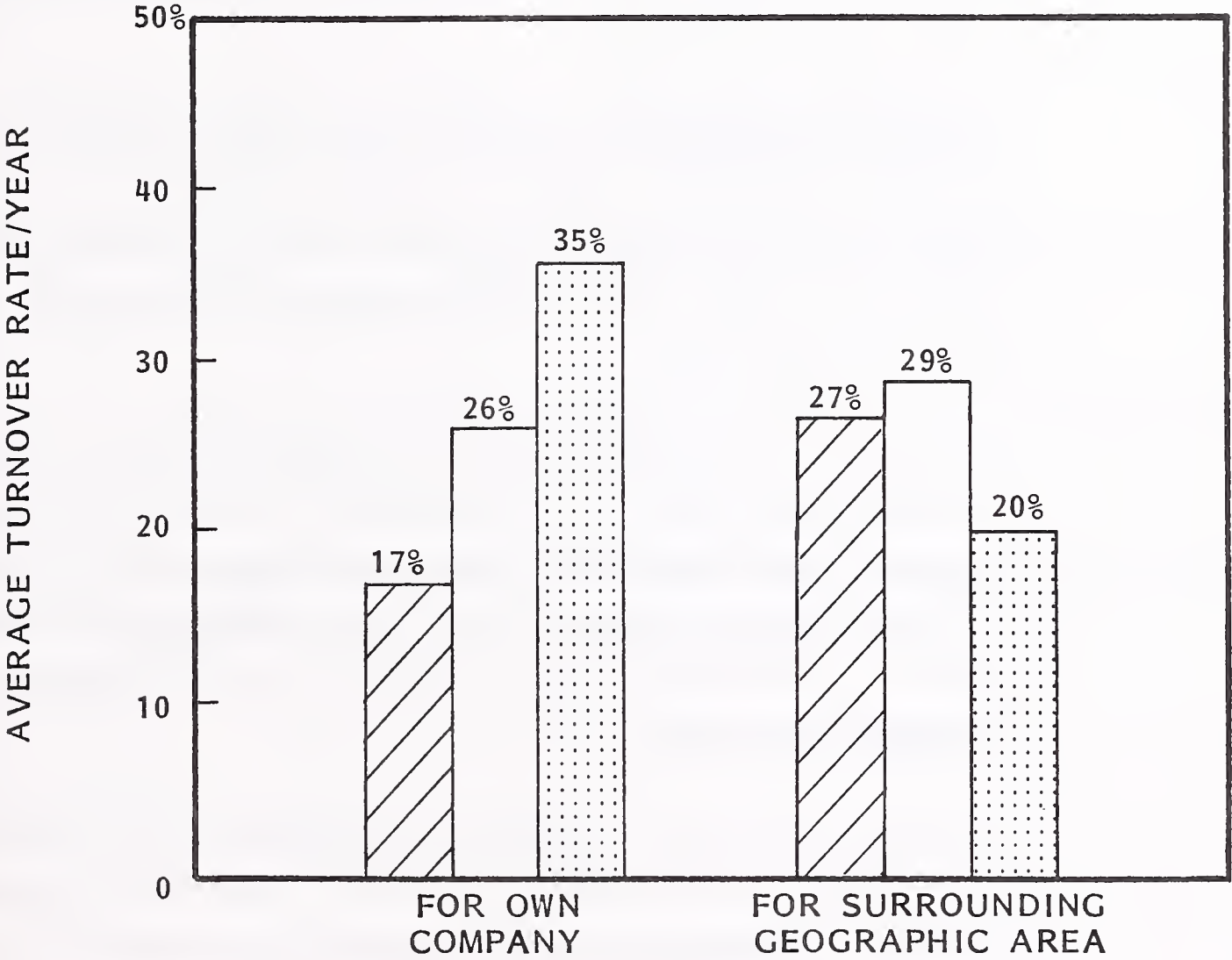





 LARGE = > \$1 BILLION REVENUES
 MEDIUM = \$100M-\$1B
 SMALL = < \$100M

SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 181

EXHIBIT III-27

EDP ANALYST/PROGRAMMER TURNOVER LEVEL, AS REPORTED BY
ON-SITE RESPONDENTS



-  MANAGEMENT'S ESTIMATE
-  PROGRAM/PROJECT LEADERS' ESTIMATE
-  PROGRAMMER/ANALYSTS' ESTIMATE

NUMBER OF RESPONSES = 27

- Limit or strictly control internal EDP department transfers so that rush jobs do not disrupt existing projects; consider hiring outside consultants rather than making disruptive internal transfers.
- One of the main barriers to planning effective strategies for decreasing turnover is the difficulty of identifying what is causing turnover in the first place. Since managers' perceptions of employees' attitudes and priorities are sometimes incorrect (e.g., see section III. B. 2), it is critical that employees' voices be heard whenever possible.
 - Good first-line supervisors have their finger on the employees' pulse and can often provide very useful, if filtered, information. To be most effective, this communication should be elicited on a structured basis, separate from other business and at regular intervals so that baseline data can be collected.
 - Open-door policies are desirable for a number of reasons, with feedback important among them. In some cases the open door can prevent turnover by releasing steam before the explosion point is reached. However, open-door data will tend to be skewed toward personality issues. It will probably not be representative of the feelings of the organization as a whole.
 - A more representative method is to ask employees for their opinions and concerns directly by means of a survey. This can be conducted by the department itself, the company personnel department or an outside organization. Anonymity is usually preferred in these surveys. Organizations with deep-seated personnel problems whose seriousness is not subscribed to by management, often find this is the only way of determining the extent of the problem.
 - Finally, when there are resignations, a structured exit interview should be held with each person to establish why they are leaving and what

actions might have kept them. Exit interviews rarely change a person's mind. They should not be approached with retention as an objective.

- Data collected from various sources should be regularly analyzed by senior DP management, with effectiveness of solutions tracked against problem areas.

5. MORALE AND MOTIVATION

- Recent studies have pointed to morale and motivation as very important factors for keeping and effectively utilizing technical staff.
- The most important motivators, according to INPUT's research, are shown in Exhibit III-28. They include:
 - Supplying a challenging learning environment (twice as important as any other factor).
 - Showing management recognition. Three entries, together about as important as the learning environment, include:
 - . Being a valued member of the organization.
 - . Receiving recognition.
 - . Receiving delegated responsibility.
 - Good general management, with well-defined goals.
- Only two factors were objectively measureable:
 - Compensation.
 - Technology employed.

EXHIBIT III-28

MOST IMPORTANT MOTIVATORS FOR ANALYST/PROGRAMMING PERSONNEL, AS REPORTED BY EDP MANAGERS

MOTIVATOR	PERCENT* OF RESPONDENTS
CHALLENGING/MEANINGFUL/LEARNING/ ENVIRONMENT	32%
BEING A VALUED, CONTRIBUTING MEMBER OF ORGANIZATION	15
STATE-OF-THE-ART TECHNOLOGY/WORKING ENVIRONMENT	15
ADEQUATE, COMPETITIVE COMPENSATION	15
GOOD SUPERVISORS/PROGRESSIVE MANAGEMENT	13
MANAGEMENT RECOGNITION/SUPPORT	12
WELL-DEFINED PROJECT/GOALS	11
DELEGATED RESPONSIBILITY	10

*MULTIPLE ANSWERS POSSIBLE
SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 142

- Other factors listed included education, flextime, fringes and working environment, as shown in Exhibit III-29.
- The single most successful factors responsible for productivity improvements are on-line development, staff participation in decision making and management support, as shown in Exhibit III-30.
- The least successful factors in productivity improvement include compensation incentives and unrealistic schedules and controls, as shown in Exhibit III-31.
- These reported results must be used with care, in that they come from organizations that have said they do not have a productivity plan and do not know how to measure productivity.
 - It would be unrealistic to expect a great surge of improved productivity from such an informal and uncoordinated approach.
 - In fact, these findings confirm the necessity of implementing a formal, integrated set of productivity initiatives that consider management, personnel, environment, users' needs and particular tools and aids.

F. A PRODUCTIVITY STRATEGY

- In the earlier sections of this chapter, the factors that are preconditions for productivity were discussed and analyzed. These factors are:
 - A commitment to quality.
 - User involvement.
 - Broad-based management.

EXHIBIT III-29

OTHER NON-SALARY INCENTIVES FOR PRODUCTIVITY IMPROVEMENT, AS REPORTED BY EDP MANAGERS

NON-SALARY INCENTIVE	PERCENT* OF RESPONDENTS
EDUCATION/COLLEGE/PROFESSIONAL SEMINARS	26%
FLEXIBLE WORKING HOURS	20
COMPREHENSIVE FRINGE BENEFITS	20
EXCELLENT PHYSICAL WORKING ENVIRONMENT	15
CAREER GROWTH	11
SPECIAL RECOGNITION/AWARDS	11

*MULTIPLE ANSWERS POSSIBLE
SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 133

EXHIBIT III-30

SINGLE MOST SUCCESSFUL FACTOR FOR PRODUCTIVITY IMPROVEMENT, AS REPORTED BY EDP MANAGERS

FACTOR	PERCENT OF RESPONDENTS
PROVIDE ON-LINE INTERACTIVE PROGRAM DEVELOPMENT	15%
STAFF PARTICIPATION IN DECISION MAKING	10
RECOGNITION FOR ACCOMPLISHMENT/ MANAGEMENT SUPPORT	10
COMPETITIVE SALARY	7
PRODUCTIVITY SOFTWARE AIDS	7
EDUCATION AND TRAINING	5
BETTER INTERACTION WITH USERS	4
IMPROVED HARDWARE AVAILABILITY	4
HIRE QUALITY PEOPLE	3
FREQUENT PROJECT REVIEW	3
SUBTOTAL	68%
OTHERS	32
TOTAL	100%

SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 138

EXHIBIT III-31

SINGLE LEAST SUCCESSFUL FACTOR FOR PRODUCTIVITY IMPROVEMENT, AS REPORTED BY EDP MANAGERS

FACTOR	PERCENT OF RESPONDENTS
SALARY/BONUS INCENTIVES	20%
UNREALISTIC SCHEDULE/DEADLINES	11
DETAILED MANAGEMENT CONTROLS	11
FLEXIBLE TIME	5
SOFTWARE PRODUCTIVITY AIDS	5
FRINGE BENEFITS	4
USER-ORIENTED LANGUAGES	4
SUBTOTAL	60%
OTHER	40
TOTAL	100%

SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 76

- Effective personnel.
- The right tools.
- The order given above is important because the factors are hierarchical; i.e., tools are not much good in the hands of ineffective workers, but effective workers are improved with the right tools.
- The most effective strategy is one that focuses on building from the base, up.
- If any of the layers of the "Productivity Pyramid" are missing, productivity will not reach the same heights and the entire productivity effort will be unbalanced.
- A commitment to quality is the foundation for all else. While users, for example, may place initial emphasis on time schedules and budgets, there is always an implicit assumption on their part that the system will meet certain quality and performance standards.
- It is the quality of the system that will determine its ultimate success or failure.
- A problem in many organizations is that those quality standards that do exist are not explicit. Consequently there is no single set of quality expectations. This results in misunderstood and misdesigned systems, with too much effort devoted to trivial systems, and not enough to more critical ones. Architectural stability is rarely considered.
- Another, more serious problem is that, in times of stress (e.g., when a project is behind schedule or over budget), quality standards may be relaxed or discarded. This occurs more easily in a setting where quality expectations have not been formalized.

- In addition, there can be no meaningful measurements of quality in a "quality-implicit" environment, since there is no agreement on what should be measured. Quality measurements are difficult, but they are the only way of setting up a positive feedback loop to improve quality.
- User involvement is rewarding, but potentially explosive. It follows upon quality of systems as an objective, and requires them to be truly effective. Otherwise, user involvement can go in cycles, with the possibility that there will alternately be:
 - A great deal of involvement and control, but disjointed and uninformed, typically with no coordination between users.
 - Little or no involvement, with the DP department running the whole show.
- Neither posture is adequate. The optimum approach - the goal of an effective DP productivity strategy - is to move to a position where users are:
 - Involved in systems development and operation.
 - Informed on what DP can and cannot do for them.
 - Aware of how their needs fit into larger company requirements.
- Broad-based EDP Management, third in the sequence of objectives, puts the needs of users and top management on an equal footing with running the DP shop. One of the biggest opportunities (and challenges) is for EDP management to educate both top management and users in non-technical DP fundamentals, including:
 - Hardware.
 - Software.

- Communications.
 - Costs.
 - Types of applications.
 - New developments.
- One cannot assume much basic DP knowledge in even the most intelligent manager. The disastrous effects of this ignorance have been the downfall of many otherwise well-intentioned systems.
 - Going beyond fundamentals, non-DP management should be engaged in issues such as:
 - Centralization versus decentralization.
 - In-house versus outside development.
 - Security and disaster recovery.
 - Productivity strategies.
 - EDP department evaluation.
 - Systems quality as a company objective.
 - Some EDP managers feel that knowledge is power. They go out of their way to accentuate the mystifying and jargonistic aspects of computers. This is a short-sighted, usually counterproductive strategy, since non-EDP management will then tend either to:
 - Do nothing most of the time.

- Make erratic, hasty decisions on DP issues.
- Make their frustrations known by eliminating the most obvious scapegoat, the EDP manager.
- Following strong management support, development of effective personnel is the key to improving productivity (narrowly defined; e.g., production rate or quantity of software). Strategies here should focus on:
 - Identification and retention of the highest-caliber personnel.
 - Motivation of all personnel, from top to bottom.
 - Skills improvement in both short- and long-term contexts, via specific course training and a broad-based career development program.
- From an operational standpoint, a single management action could lead to improvements in many areas. For example, acquiring an in-house video training system could improve skills, motivation and satisfaction, which would in turn reduce turnover.
 - It should be stressed again that even the most highly skilled, dedicated personnel will not achieve productivity from a long-term, corporate standpoint unless they are working on quality systems that will serve important corporate needs.
- Capping the "productivity pyramid" is the use of the right tools. They can be a very important asset in achieving "micro" productivity, if they:
 - Are consistent with employee skill levels.
 - Fit together in an integrated package.
 - Focus on the organization's critical areas of need and concern.

- Are viewed as a help, not a hindrance, to those who will use them.
- Are appropriate for the stage of organizational development of the DP organization. (For example, an EDP department that discourages user participation in the development process will not find a user-oriented query package very helpful.
- There is no single "cookie cutter" strategy useful for all organizations.
 - The stage of development of a particular organization will greatly influence which approaches are feasible. (See Chapter IV and Appendix E).
 - Also important are the "Corporate Culture" characteristics of an organization:
 - Organizational placement of EDP.
 - Degree of centralization in the organization as a whole.
 - The personality characteristics of key EDP and non-EDP managers.
- An effective strategy, in the final analysis, is one that is tailored to the particular organization.
- Successful strategies take time to achieve. The planning and implementation of successful strategies can, and often should, take years until all the major pieces are in place. Of course, much work can be done immediately, but such areas as user involvement can take a long time to achieve.
 - This time factor should be taken into account in a very realistic and cold-blooded way. Organizations that are in a constant state of flux place value mainly on short-term results. In that kind of setting, it

would be a waste of resources to attempt many of the more important productivity initiatives. However, the best is sometimes the enemy of the good.

G. MANAGEMENT BY OBJECTIVES: A FIRST STEP

- Management by Objectives (MBO) is a management technique whose major components include:
 - The joint setting of objectives by a manager and subordinate.
 - A plan detailing how objectives will be reached.
 - The resources that the subordinate requires.
 - The resources that the manager agrees to make available, and a time table outlining when they will be made available.
- Objectives, in the MBO context, are for a specified time period (e.g., 6 or 12 months), and should be quantifiable or measurable (e.g., not "reduce errors significantly," but "reduce errors 25%").
- MBO will not be successful unless it is installed from the top down in an organization (or organizational component) that wants to use it.
- MBO has proven very useful in companies that have installed it, because it:
 - Allows a multidimensional measurement of activities, which can still be related to P&L (where profit and loss is appropriate).
 - Clarifies communications and responsibilities.

- Improves morale and motivation.
- An MBO system could be installed in the EDP department alone, although it would be somewhat more effective and better understood if it were part of a corporate or divisional effort.
- An effective MBO program should concentrate on:
 - Improving morale and motivation.
 - Focusing training efforts to make them more effective.
 - Producing an inventory and improvement of skills.
 - Improving productivity at the micro level by establishing productivity targets.
 - Reducing turnover by improving the quality of worklife.
- On the cautionary side, it should be noted that MBO programs are difficult to install and "make stick".
 - The timeframe to install is typically measured in years, since MBOs proceed down through the organization, layer by layer. A manager cannot effectively set up his or her own MBO program while at the same time setting up programs for subordinates.
 - A good MBO program takes a great deal of management time. This is time that cannot be delegated.
 - Quantification, though imperative, may be resisted, and targets may be made too easy. (Almost as often, they are made too hard - usually by the subordinate!)

- The interface between MBO and a performance or salary evaluation has to be carefully studied to see what the best relationship is for the particular organization. Sometimes a performance evaluation system continues as a totally separate, unchanged program, but sometimes it is totally absorbed. There have been successes and failures using both approaches.
- For more information, the best source is another company that has already established an MBO program. A "do-it-yourself" approach is harder and less likely to succeed.

IV THE PRODUCTIVITY SELF-ANALYSIS MATRIX

IV THE PRODUCTIVITY SELF-ANALYSIS MATRIX

A. STAGES OF PRODUCTIVITY DEVELOPMENT

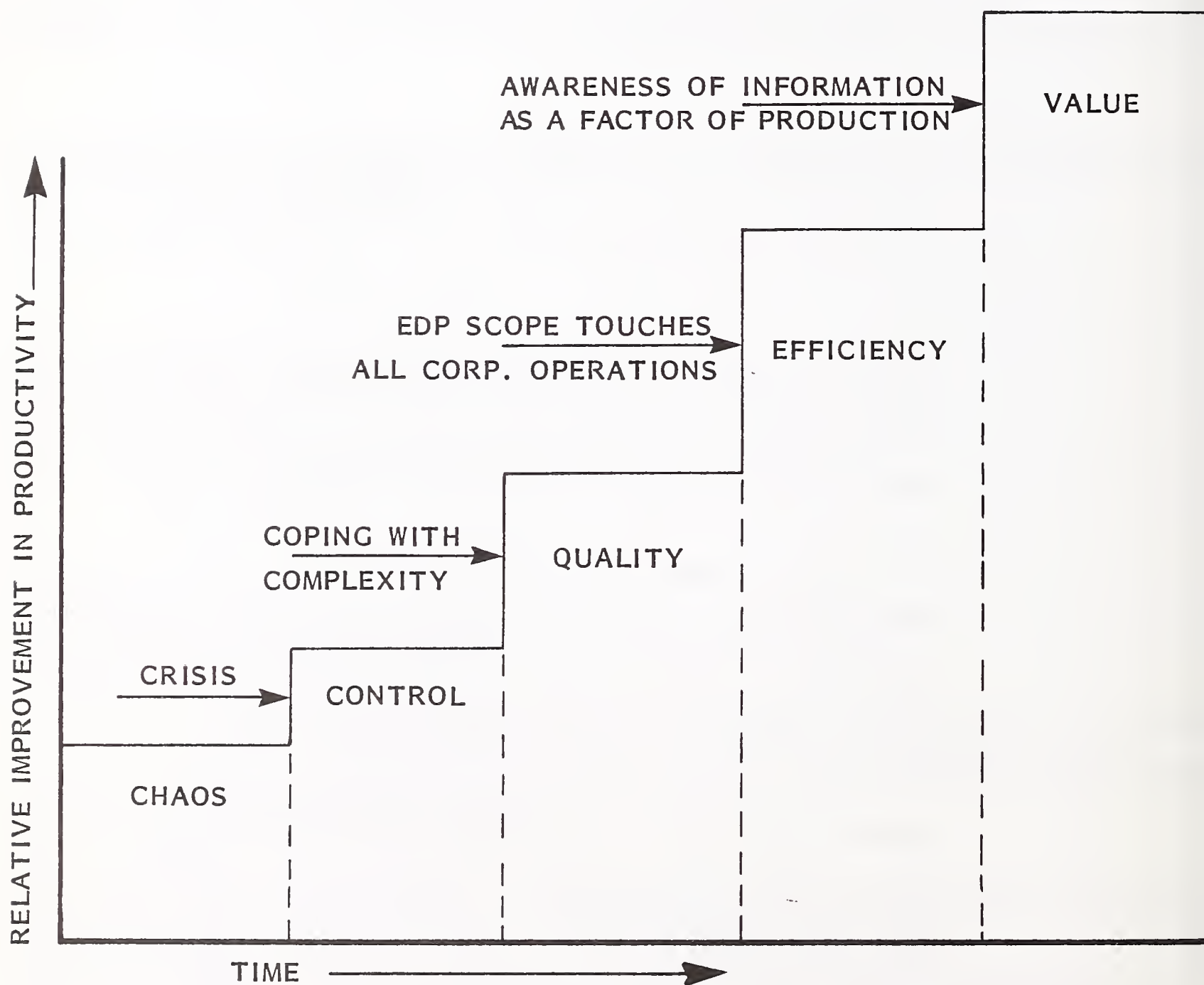
- Key to a successful software productivity strategy is the choice of individual productivity initiatives that are consistent with a data processing organization's current stage of development.
 - If a particular productivity initiative is too far ahead or behind what the DP department is doing in other areas, not only may the initiative fail, but it may also be counterproductive.
 - In any case, it is unlikely to achieve the desired level of improvement.
- On-site interviews with the participating organizations in this study revealed learning curves both for individual techniques and for productivity improvement as a whole. These learning experiences had an influence not only upon how far the organization had gotten in its improvement efforts, but also upon how fast it was able to proceed and how many times it had to start over.
- These experiences, taken together, imply that there are certain fundamental attitudes, organizational structures and other characteristics that interact or cooperate to define in a concise, but generalized, way the level of productivity achievement of the organization.

- In recent years, "stages of organizational growth" have been a popular management topic. (See, for example, Richard L. Nolan's article on "Managing the Crises in Data Processing" in the March-April 1979 issue of the Harvard Business Review.)
- The usefulness of the approach is that it introduces the concept of progression in an organization's development.
 - This progression is often mirrored in the type and complexity of productivity strategies used at different stages of organizational development.
- From the standpoint of productivity, there appear to be five definable stages in DP development.
 - Stage 0: Chaos.
 - Stage 1: Control.
 - Stage 2: Quality.
 - Stage 3: Efficiency.
 - Stage 4: Value.
- In each stage, different productivity strategies are called for. In the later stages, the productivity initiatives are often more complex and difficult, but the payoff can be very high.
- Note that while these five stages do not, and are not intended to, correspond in some definite way to Nolan's six stages (which basically address a more general set of interests), they do point out certain relationships to the maturity of an organization's development and focus.

- These relationships will be reflected in the relative weights and point values attached to the use (or absence) of certain managerial tactics and productivity aids, particularly resulting from their appropriateness at a given stage of development.
- The stages of development are not arbitrary or isolated, but have an organic relationship that comes from successive motivating factors, as shown in Exhibit IV-1.
 - Chaos is self-descriptive: a constant state of crisis that demands efforts to bring it under control.
 - After control has been achieved, the organization realizes that the semi-arbitrary mechanisms used to establish control must be modified to take into account the twin demands for increasingly complex systems and higher-quality systems.
 - Quality DP systems raise interest throughout the organization in the potential for DP to increase the efficiency and effectiveness of the entire organization. Many of the productivity initiatives begun in the quality stage are refined and expanded in the efficiency stage.
 - The efficiency stage raises further expectations within the organization, in that user needs for information are met more predictably, and information in and of itself is viewed as having value alongside the other factors of production (labor, capital, materials).
 - In the value stage, data processing is no longer seen as a separate, isolated activity but participates directly in the mainstream of corporate activities (as, for example, finance does now). It is in the value stage that truly large gains in productivity can be achieved, as users interact directly with the information needed to expand the market control and profitability of the entire organization.

EXHIBIT IV-1

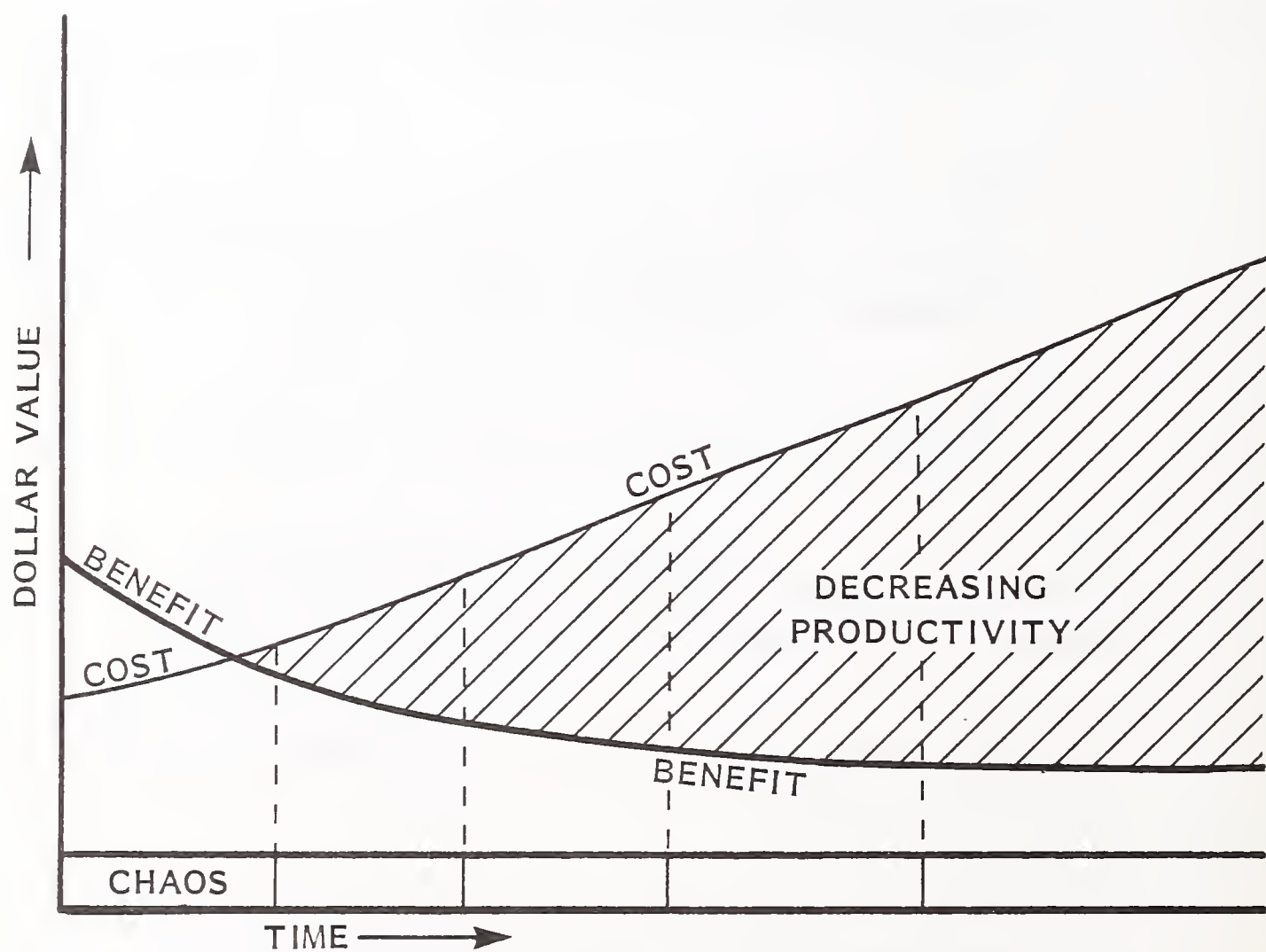
MOTIVATING FACTORS IN MOVING FROM STAGE
TO STAGE IN PRODUCTIVITY IMPROVEMENT



- Few organizations today have entered the value stage and, unfortunately, many are only now beginning to emerge from the chaos stage.
- A forecast of relationships between EDP costs and EDP benefits, as seen by the organizations in stage zero, looks depressingly like Exhibit IV-2. Costs continue to rise with no end in sight, while benefits (at least as perceived by corporate financial management) are declining.
- At this stage, there is a strong temptation to institute EDP improvements directed at efficiency; but these have rarely, if ever, achieved their desired result.
 - There is no standard against which to measure any benefits that may be achieved.
 - There is no basis of support for any of the new techniques to build on, but there is a high element of risk in trying unfamiliar techniques on critical projects.
 - Frequently the end of this stage is marked by a change of DP management.
- For many of the same reasons, it is not appropriate at this stage to attempt to institute the whole range of quality improvements. In addition:
 - The payback period from quality improvements lies too far in the future to be impressive.
 - The quality improvements have a generally steep learning curve, which would initially result in delaying, rather than speeding up, the delivery of systems.
- The situation calls for the implementation of firm control procedures, including centralized approval of major systems, improved coordination of

EXHIBIT IV-2

STAGES OF PRODUCTIVITY IMPROVEMENT:
COST/BENEFIT RELATIONSHIPS, STAGE 0



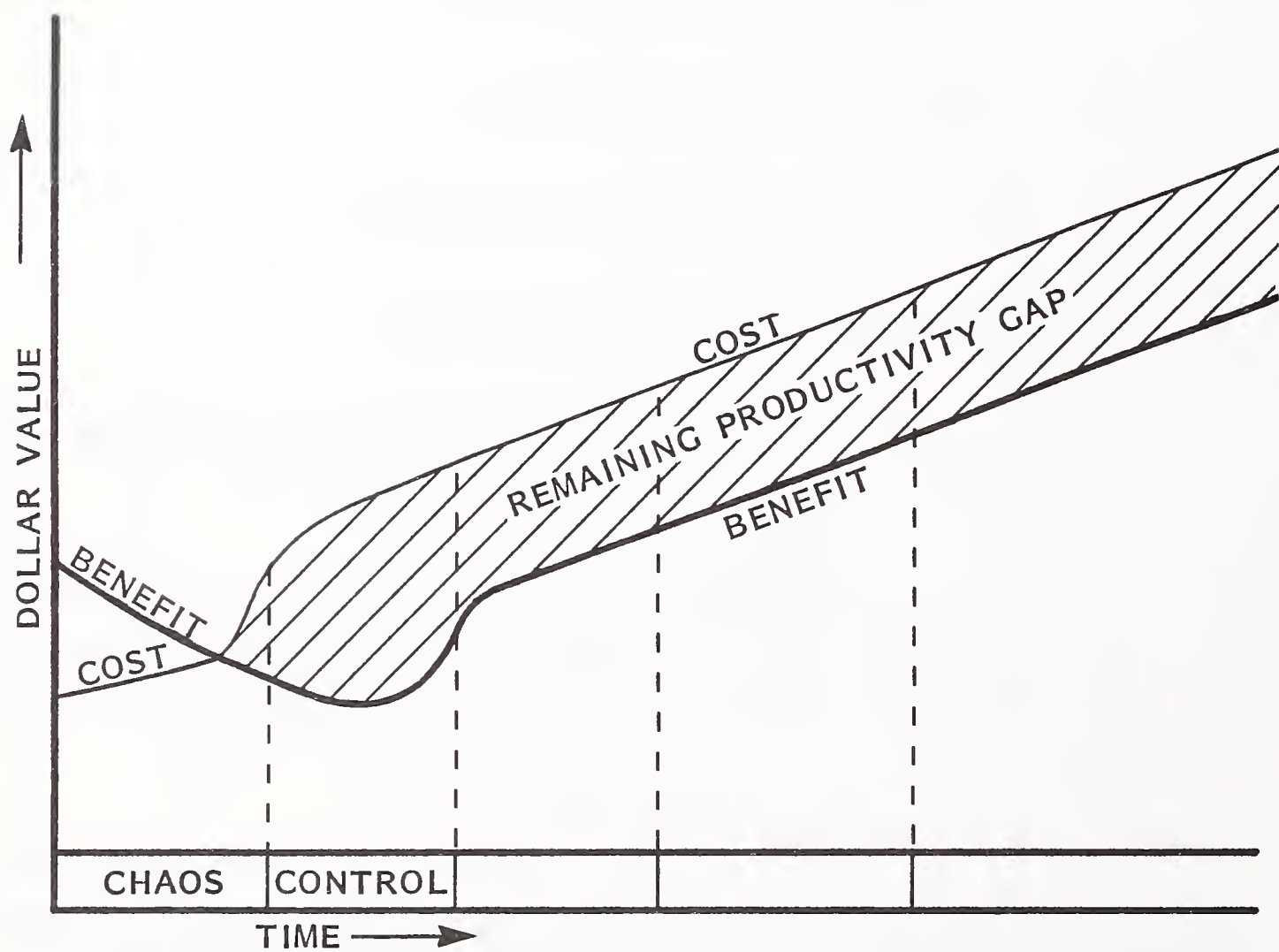
- BENEFIT LINE, WHERE CHAOS CONTINUES OVER TIME.
- COST LINE, WHERE CHAOS CONTINUES OVER TIME.

requirements, and some initial attempts to plan the development of information systems rather than respond to random user "needs."

- Exhibit IV-3 presents a typical scenario for stage one, in that the downward trend of the benefit line turns perceptibly upward.
- However, some additional costs are incurred in establishing control, because additional functions must be performed, including:
 - . Preparation of budgets and schedules.
 - . Approval of budgets and schedules.
 - . Development of procedures and standards.
 - . Documentation of anticipated benefits.
- The net result is a continuing (although smaller) gap between costs and benefits that cannot be closed by means of controls alone.
- However, the foundation has been laid for determining standards, measuring future results and instituting improvement strategies that will lead future benefits over the cost line to show positive dollar returns.
- An attempt to institute the efficiency measures that were inappropriate earlier has a higher probability of success at this point, but should nevertheless be deferred until after certain quality initiatives have been implemented.
- Otherwise the efficiency initiatives will be short term (at best), uncoordinated and likely to create additional problems faster than they are solved.

EXHIBIT IV-3

STAGES OF PRODUCTIVITY IMPROVEMENT:
COST/BENEFIT RELATIONSHIPS, STAGE 1



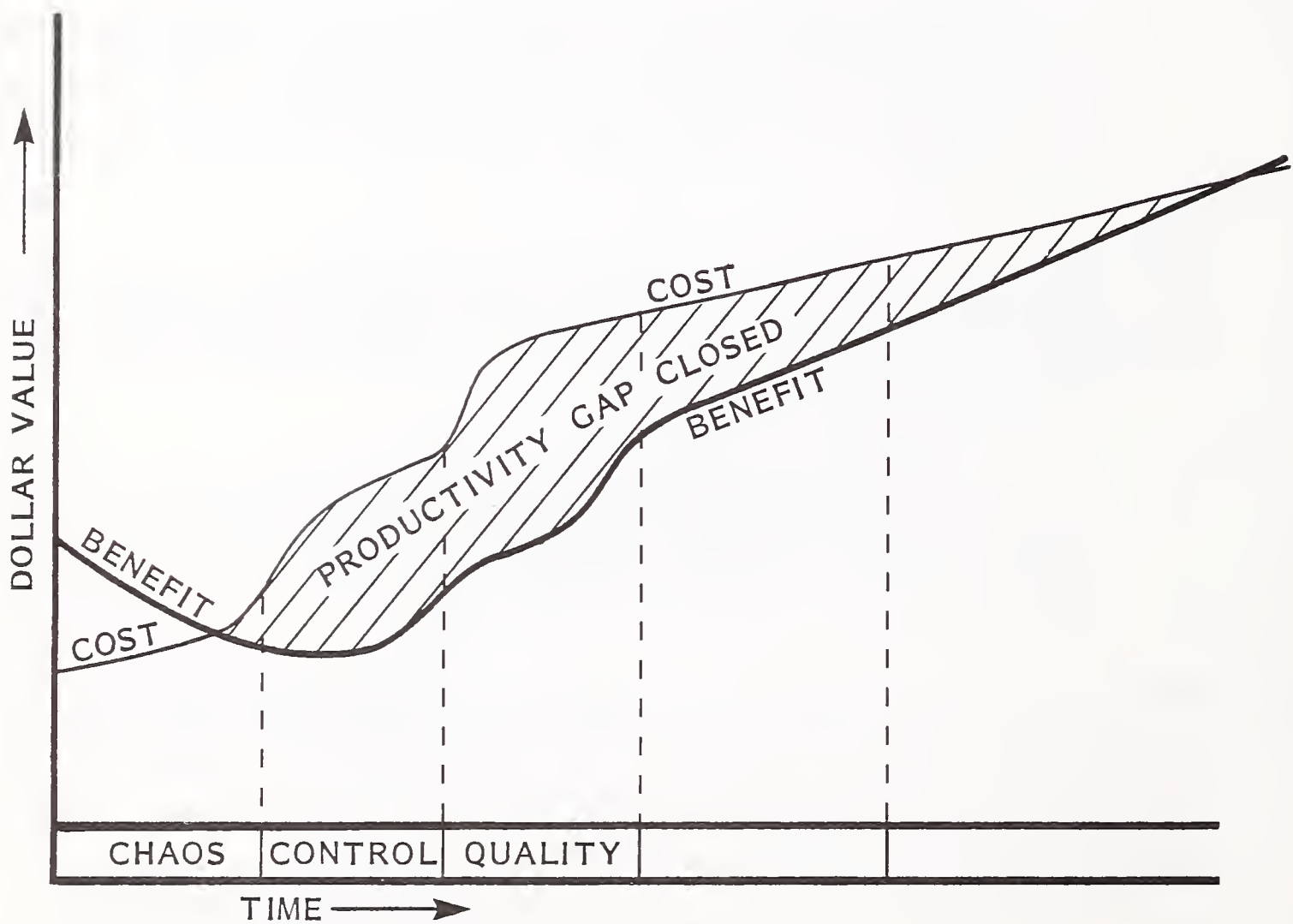
—— BENEFIT LINE, WHERE CONTROL ONLY IS INSTITUTED OVER TIME.

—— COST LINE, WHERE CONTROL ONLY IS INSTITUTED OVER TIME.

- The preferred next stage is to institute quality initiatives, continuing to build upon the control strategies begun in stage one, and eventually closing the productivity gap, as illustrated in Exhibit IV-4.
- This will be the most difficult stage for most organizations to achieve, because:
 - Significant changes are required in the EDP organization.
 - A considerably higher level of skill and experience is demanded to cope with the increased complexity of data base systems, communications and integration of diverse user groups with frequently conflicting requirements.
 - The testing and quality assurance requirements impose another significant delay (at least initially), and are a psychologically difficult "sell" to the end user.
 - Taken together, the dollar costs of this stage are large and front-loaded, representing a continuation of investment where the eventual payback may be years in the future.
- Now is the time to institute the efficiency initiatives that were deferred earlier, both to shorten the payback time horizon displayed in Exhibit IV-5, as well as to firm up the foundational mechanisms that have been put in place to put the user in direct interaction with the company's information resources.
 - Stage three will be a relatively short, transitional stage between quality and value if the EDP management team has done an effective job of implementing the quality initiatives of stage two.
 - If stage two was not properly motivated and assured, it is doubtful whether even stage two can be sustained.

EXHIBIT IV-4

STAGES OF PRODUCTIVITY IMPROVEMENT: COST/BENEFIT RELATIONSHIPS, STAGE 2

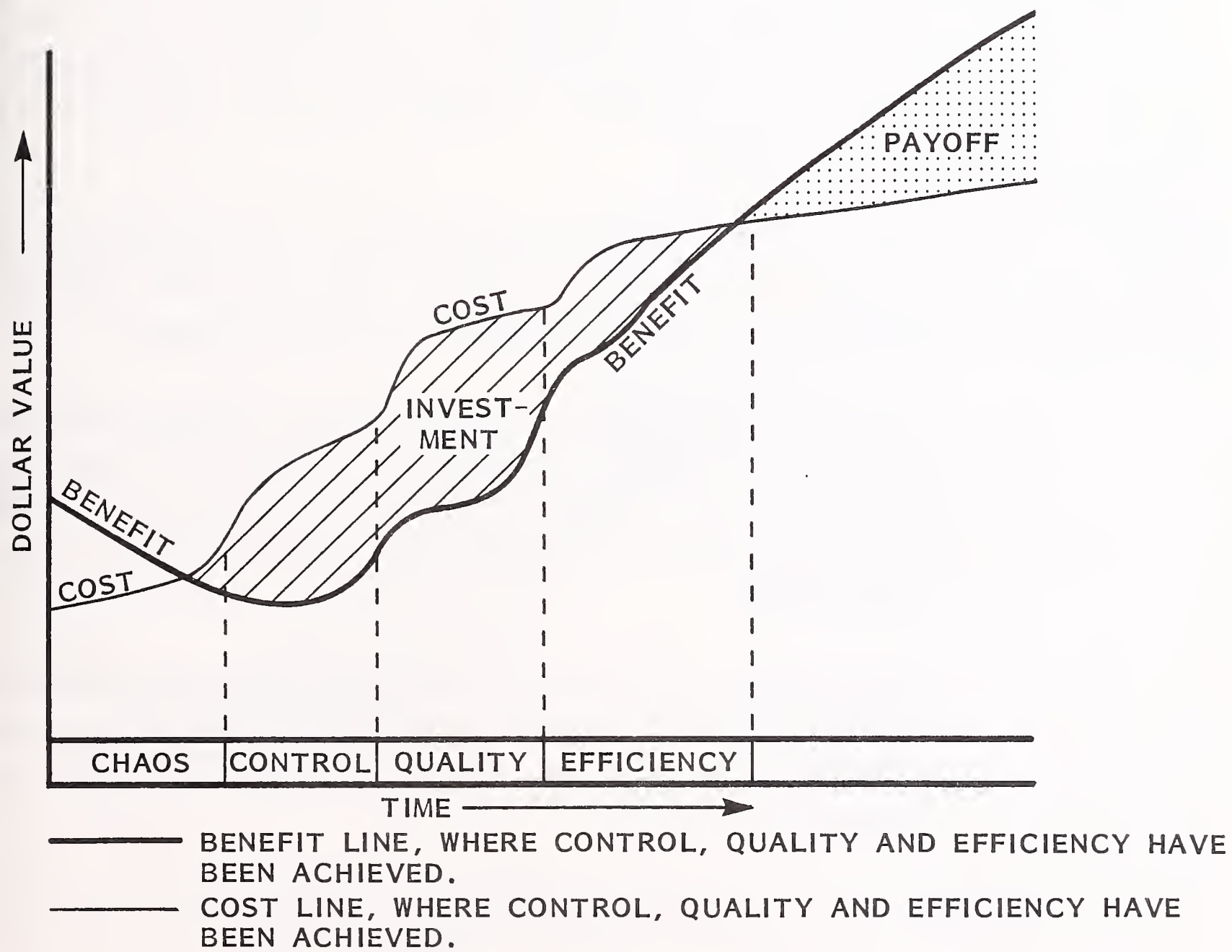


— BENEFIT LINE, WHERE CONTROL AND QUALITY HAVE BEEN INSTITUTED.

— COST LINE, WHERE CONTROL AND QUALITY HAVE BEEN INSTITUTED

EXHIBIT IV-5

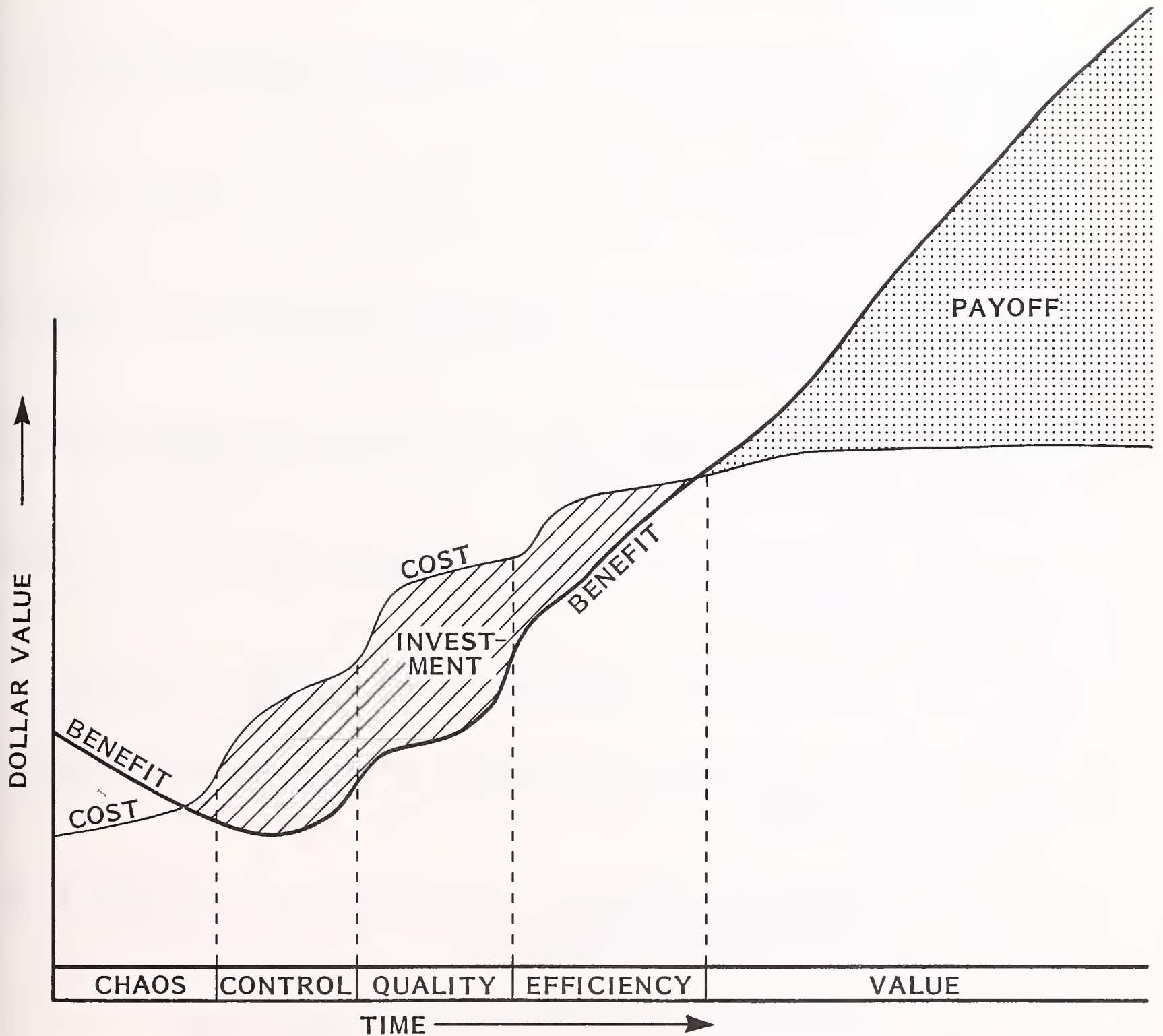
STAGES OF PRODUCTIVITY IMPROVEMENT:
COST/BENEFIT RELATIONSHIPS, STAGE 3



- More likely, enough "modifications" and "exceptions" will be made to the system development methodology and structured design approach to deprive them of their value and leave only the formalities and overhead associated with their use.
- Eventually even these will disappear and the organization will return to stage one.
- Organizations that have entered stage three and have been able to implement its strategies throughout the organization will find stage four a very natural step. This stage, as exemplified by Exhibit IV-6, provides a very large payoff for very little increased expenditure, and easily repays the investment of the preceding three stages.
 - The key to achieving payoff in this stage, and to maximizing the return on investment, is the degree to which control and access to strategic business information has been placed in the hands of the users.
 - Compared to the productivity gains accrued from making the "process" of data processing more efficient through stages one to three, the productivity gains in stages three and especially four are enormous, because the information produced in these stages makes the end user more effective.
 - At the same time, the process of generating information in stage four gains from efficiency advances currently being made in integrated programming and design tools, and improved theories of testing and certification of reliability.

EXHIBIT IV-6

STAGES OF PRODUCTIVITY IMPROVEMENT: COST/BENEFIT RELATIONSHIPS, STAGE 4



- BENEFIT LINE, WHERE CONTROL, QUALITY, EFFICIENCY AND VALUE HAVE BEEN ACHIEVED
- COST LINE, WHERE CONTROL, QUALITY, EFFICIENCY AND VALUE HAVE BEEN ACHIEVED

B. THE CHARACTERISTICS OF EACH STAGE OF SOFTWARE PRODUCTIVITY DEVELOPMENT

- Each stage has its own set of distinguishing characteristics. Some of the more important factors include:
 - EDP organization: How is the DP function organized to carry out its responsibilities?
 - Application type: What are the characteristics of application systems themselves?
 - EDP management role: How are its role and responsibilities viewed in the organization?
 - Control mechanism: What is the structure and procedure used to set priorities and make plans?
 - Performance characteristics: Are projects done on time and within budget?
 - Quality attitude: How important is quality, and how is it achieved?
 - User involvement: To what extent are users involved, and how are they involved?
 - Programming resources: How do programmers get access to the computer for development work?
 - Productivity tools: What kinds of support tools are used?
 - Investment payoff: What is the balance between costs and benefits?

- Stage zero ("Chaos") is typified, in Exhibit IV-7, by user "ownership" of particular applications systems (which are usually batch). EDP management tries to balance the user power centers in order to assume effective control.
 - There is no effective quality strategy other than to hope that systems initially appear to work. Maintenance is unending, in large part because the controlling users are constantly demanding system changes.
 - High-level languages are the only "productivity" tool, and holdover attitudes from an earlier generation are likely to favor operational efficiency over flexibility and ease of management.
 - Costs and dissatisfaction are high.
 - In summary, while favored users are often happy, other users and top management grow increasingly restive.
- Stage one ("Control") emerges as a reaction to "Chaos," as shown in Exhibit IV-8. Users are almost totally excluded when the DP organization is centralized.
 - Applications are integrated (e.g., divisional payroll systems and departmental charts of accounts will be unified); formal DP budgets and schedules become important as DP management strives to attain control over projects and costs.
 - A start is made on providing programmers with better development tools (e.g., TSO), but associated costs are frequently frightening.
 - Quality control is still vestigial and primarily defensive (e.g., separation of test and production versions of programs).
 - Unfortunately, the emphasis on control often results in even slower development, and the backlog of unfinished work increases.

EXHIBIT IV-7

CHARACTERISTICS OF STAGE ZERO: CHAOS

EDP ORGANIZATION	APPLICATION TYPE	EDP MANAGEMENT ROLE	CONTROL MECHANISM	PERFORMANCE CHARACTERISTICS
BUSINESS FUNCTION "OWNED" BY USER	FUNCTIONAL COST REDUCTION, BATCH	PAWN; FOCUS ON OPERATIONS, WORKING TOWARD CONTROL	VERBAL OR SIGNED PROJECT REQUEST	SLOW

QUALITY CONTROL	USER INVOLVEMENT	PROGRAMMING RESOURCES	PRODUCTIVITY TOOLS	INVESTMENT PAYOFF
"IT WORKS"	REQUESTOR, DIRECT RELATIONSHIP ("OWNER")	BATCH/RJE	COBOL, FORTRAN, PL/1, BASIC, ASSEMBLER (!)	UNCONTROLLED COSTS

EXHIBIT IV-8

CHARACTERISTICS OF STAGE ONE: CONTROL

EDP ORGANIZATION	APPLICATION TYPE	EDP MANAGEMENT ROLE	CONTROL MECHANISM	PERFORMANCE CHARACTERISTICS
CENTRALIZED BUSINESS FUNCTION	INTEGRATION OF OPERATING DIVISIONS	MANAGER; FOCUS ON COORDINATION, WORKING TOWARD QUALITY	BUDGET, SCHEDULE	COST OVERRUNS, SLOWER, BACKLOG GROWING

QUALITY CONTROL	USER INVOLVEMENT	PROGRAMMING RESOURCES	PRODUCTIVITY TOOLS	INVESTMENT PAYOFF
PRODUCTION VERSION VERSUS TEST VERSION	"LOCKED OUT"	RJE, SOME ON-LINE TERMINALS	TSO OR EQUIVALENT	BEGINNING TO CONTROL COSTS, CONSCIOUS INVESTMENT IN IMPROVEMENT TECHNIQUES

- Aggrieved users in the largest divisions often succeed in wresting back control and operation of DP. Particularly unfortunate organizations may oscillate between stages zero and one.
- Stage two ("Quality") represents a better resolution of the user-DP tension, where the user is slowly reintroduced into the software development process and takes part in the steering committee control mechanisms, as shown in Exhibit IV-9.
 - Applications become increasingly complex in this stage, with cross-functional integration and on-line data bases coming into vogue. A matrix organization begins to emerge within the DP department, although it probably will not be claimed as such.
 - Quality becomes a very important concern to keep such complex systems operating, and usually a greatly expanded testing function is created, including test planning (with or without a separate quality assurance group) and a defined testing phase.
 - The most important initiative in this stage (in terms of its impact on later stages) is the implementation of a formal systems development methodology. Productivity tools also become important, especially those which support quality (structured design, macro languages).
 - Programmers are given easier access to the computer.
 - The investment for all of these improvements rises at a higher rate than the benefits, but on-time delivery eventually becomes the rule, and the backlog of undone work stabilizes at a level that can be predictably achieved within one to two years.
- Stage three ("Efficiency") represents a gradual transition from stage two to stage four.

EXHIBIT IV-9

CHARACTERISTICS OF STAGE TWO: QUALITY

EDP ORGANIZATION	APPLICATION TYPE	EDP MANAGEMENT ROLE	CONTROL MECHANISM	PERFORMANCE CHARACTERISTICS
TECH SPECIALIST MATRIX PLUS BUSINESS FUNCTION, PROJECT BASIS	CROSS-FUNCTIONAL INTEGRATION, DATA BASE, ON-LINE	DIRECTOR; FOCUS ON PLANNING, WORKING TOWARD EFFICIENCY	FORMAL SDM, STEERING COMMITTEE	BACKLOG STABILIZED, 80% ON TIME

QUALITY CONTROL	USER INVOLVEMENT	PROGRAMMING RESOURCES	PRODUCTIVITY TOOLS	INVESTMENT PAYOFF
SPECIFIC TEST PLAN, SEPARATE TEST PHASE AND QUALITY ASSURANCE GROUP	RELUCTANT PARTICIPANT, EARLY STAGES OF PROJECT	CLUSTERED OR SHARED TERMINALS, ASSURED DEVELOPMENT ACCESS	STRUCTURED DESIGN TECHNIQUES, DATA BASE ORIENTED LANGUAGES	CONTINUED INVESTMENT

- Users are much more integrated in the planning and development process.
- The DP matrix organization evolves further as users take an active, collaborative role in system development.
- Data bases become enormous and support very specialized, as well as strategic, applications. Advanced users begin to get direct access to the data base.
- Automated tools developed by the EDP organization are somewhat integrated, both to facilitate use and, more importantly, to build quality, testability and maintainability into the design process by eliminating the need for human intervention wherever possible.
- Most projects are now finished on time and within budget; backlog is no longer an issue. This reflects both the greater degree of skill attained in predicting requirements and the greater appreciation of precisely specifying requirements before proceeding further.
- Exhibit IV-10 encapsulates the description of stage three.
- Stage four ("Value") closes the circle, in a sense, since the user is again dominant, but in a totally different way than in stage zero. Exhibit IV-11 describes this stage.
 - Distributed user applications are constructed within a common framework; users routinely access the data bases directly and perform much "programming" with user-oriented tools.
 - In some cases users will provide development leadership; in general, the distinction between user and EDP is blurred as they become partners in a common endeavor.

EXHIBIT IV-10

CHARACTERISTICS OF STAGE THREE: EFFICIENCY

EDP ORGANIZATION	APPLICATION TYPE	EDP MANAGEMENT ROLE	CONTROL MECHANISM	PERFORMANCE CHARACTERISTICS
MATRIX, TECH SPECIALIST AND BUSINESS SPECIALIZATION (USER MANAGER)	LARGE-SCALE DATA BASE, ON-LINE, SPECIALIZED AND STRATEGIC ORIENTATION	COMPANY OFFICER, CONCERNS BEYOND EDP; WORKING TOWARD VALUE	HIERARCHICAL STEERING COMMITTEES, LONG-RANGE PLAN, PRODUCTIVITY MEASUREMENTS	ON TIME, COST WITHIN BUDGET

QUALITY CONTROL	USER INVOLVEMENT	PROGRAMMING RESOURCES	PRODUCTIVITY TOOLS	INVESTMENT PAYOFF
AUTOMATED TESTING, RELIABILITY A DESIGN ISSUE	PARTICIPANT THROUGHOUT PROCESS	TERMINAL PER PROGRAMMER AND ANALYST	AUTOMATED, INTEGRATED SET OF TOOLS, LIMITED DIRECT USER ACCESS	EMERGING PAYOFF

EXHIBIT IV-11

CHARACTERISTICS OF STAGE FOUR: VALUE

EDP ORGANIZATION	APPLICATION TYPE	EDP MANAGEMENT ROLE	CONTROL MECHANISM	PERFORMANCE CHARACTERISTICS
DISTRIBUTED; CENTRAL GROUP DOES COMMON SYSTEMS, TECHNICAL ADVISORY	"WHAT IF" AND RETRIEVAL BY USER. CENTRAL GROUP DOES TECHNICAL SYSTEMS AND MAINTENANCE OF STRATEGIC DATA BASE RESOURCE	OPERATING MANAGEMENT OF COMPANY; FOCUS ON SURVIVAL, GROWTH AND PROFITABILITY OF COMPANY	PRODUCTIVITY INDEX, VALUE INDEX	ASSUMED, NOT AN ISSUE

QUALITY CONTROL	USER INVOLVEMENT	PROGRAMMING RESOURCES	PRODUCTIVITY TOOLS	INVESTMENT PAYOFF
QUALITY ASSURANCE A MAJOR DESIGN OBJECTIVE	ACTIVE LEADER, DOES SOME OF OWN SYSTEMS	SPECIALIZED, INTEGRATED DEVELOPMENT TERMINALS	WIDESPREAD DIRECT USER ACCESS, SELECT SOFTWARE BUILDING BLOCKS	SUBSTANTIAL PAYOFF

- Quality assurance and maintainability become the primary design objectives (e.g., often achieved through pretested software modules or building blocks).
- Data processing produces a significant payoff to the organization, at long last delivering on its promised benefits of increased marketing power, customer service and greater corporate profitability through strategically optimized applications of information.

C. IDENTIFYING PRODUCTIVITY DETERMINANTS, INSTRUMENTS AND RESULTS

- The characteristics of each stage addressed above can be divided into three types:
 - Those that are a precondition or determinant of productivity. In the short run, these are givens that cannot easily be changed.
 - Those that are a means or instrument to achieving increased productivity.
 - Those that show the effects or results of productivity.
- In the descriptions that follow, the status of each characteristic is shown in each organizational stage.
 - For certain characteristics these are actions that the EDP department can take to improve conditions, perhaps moving to a higher stage earlier than would otherwise be possible.

I. DETERMINANTS OF PRODUCTIVITY

- Application type: The progression shown below is typical for most organizations' applications. The importance of this sequence from a productivity standpoint is that the more advanced productivity aids are often of marginal value for less complex DP systems. Trying to institute these aids too early will frequently be wasteful.
 - Stage zero: Batch, functional cost reduction applications.
 - Stage one: Integration of operating divisions.
 - Stage two: Cross-functional integration, data base, a few on-line applications.
 - Stage three: Large-scale data base; many on-line, specialized and strategically orientated applications.
 - Stage four: "What-if" retrieval done by user; central EDP group does technical systems and maintenance of the strategic data base resource.
- EDP management role: The role of EDP management mirrors the attitude of the overall organization. If the organization itself does not value planning and cooperation, then it is doubtful whether EDP can progress to higher levels of productivity.
 - Stage zero: Head of EDP seen as a pawn; the EDP focus is on operations, working toward control.
 - Stage one: Head of EDP seen as a manager; focus is on coordination, working toward quality.

- Stage two: Head of EDP seen as director; focus is on planning, working toward efficiency.
- Stage three: Head of EDP seen as a company officer; focus lies beyond EDP; working toward value.
- Stage four: Head of EDP serves on corporate operating committee; focus is on the survival, growth and profitability of the entire corporation.

2. INSTRUMENTS OF PRODUCTIVITY

- EDP organization: The way in which EDP is organized, especially the way personnel issues are dealt with, can have a profound effect on EDP productivity and effectiveness. As the personnel structure becomes more complex (matrix management and technical specialization) it becomes even more important that environmental issues affecting morale, turnover, etc., be handled satisfactorily.
 - Stage zero: The EDP group is organized by business function, with each group of programmers and analysts "owned" by their user.
 - Stage one: Business function groups are centralized.
 - Stage two: Each business function is now organized on a project basis, with technical specialists available to projects on a matrix basis.
 - Stage three: The whole EDP group is organized as a Matrix. Projects are organized by business specialization, often with a user manager. The technical specialization matrix remains in place.
 - Stage four: The EDP function is largely distributed, but a central group continues to serve in a technical advisory role while doing common systems of high complexity.

- Control mechanisms: Control mechanisms begin as simple devices that are often ineffective or crude, move to a more complex committee structure and culminate in sophisticated measurements of productivity and effectiveness. Work on the measurement process is still in a very preliminary state through stages zero to two. If begun too soon, it has a counterproductive effect.
 - Stage zero: Verbal or signed project request is usually sufficient.
 - Stage one: Budget and schedule are required.
 - Stage two: Formal system development methodology and steering committee are implemented.
 - Stage three: Hierarchical steering committees, long-range plans and productivity measurements become useful.
 - Stage four: Productivity index and value index are essential.
- Quality attitude: In the early stages (0-1), quality is not stressed because it is a longer-term issue. In the later stages (2-4), it becomes clearer that quality will decide whether DP can supply the information that will become central to the organization.
 - Stage zero: "It works" describes the primitive level of appreciation for quality control. In fact, the systems implemented at this stage show their quality by accident, not by design.
 - Stage one: Production version versus test version of files and programs are defined, and (hopefully) eliminate most of the catastrophic effects of poor quality control.
 - Stage two: Specific test plan, separate test phase and a designated quality assurance group are established.

- Stage three: Automated testing and reliability become design issues.
- Stage four: Quality assurance is a major design objective. Robustness is critical to support large-scale user access.
- User involvement: User involvement is cyclical. Initially, the user is in "control," although the user is almost always unable to exploit more than a fraction of the potential of the computer. To regain technical control, the user is excluded by the EDP organization in stage one. When the user does again become fully involved, it will be as an equal, knowledgeable partner who understands the value of information as a resource.
 - Stage zero: User, as the requestor, exercises a direct relationship ("owner") with "his" systems and "his" programmer.
 - Stage one: User is virtually "locked out" by the EDP organization.
 - Stage two: User returns as a reluctant participant in the early stages of the project.
 - Stage three: User becomes a strong participant throughout the process of system development.
 - Stage four: User functions frequently as an active leader, doing some of its own systems.
- Programming resources: The availability of programming resources is a clear continuum of increasing programmer/computer interaction.
 - Stage zero: Exclusively Batch or RJE access.
 - Stage one: Predominantly RJE, but some on-line terminals are available (primarily to systems programmers).

- Stage two: Clustered or shared terminals, up to a ratio of 1 terminal per 4 programmers. At the very least, development access is always assured (not preempted by production overruns).
 - Stage three: Normally a terminal is available for each programmer and analyst.
 - Stage four: Specialized, integrated development terminals are in widespread use, providing built-in, integrated software support for analysts as well as programmers.
- Productivity tools: The core of the "tool-building" effort occurs during stages two and three. Providing effective tools is a key for maximizing the potential of these stages, and for advancing to later stages. Providing them earlier is rarely effective, because the potential of the tools is not understood.
- Stage zero: COBOL, FORTRAN, PL/I, BASIC. In a usually misguided attempt to optimize resources, ASSEMBLER is sometimes used as a "productivity tool"!
 - Stage one: TSO or equivalent makes an obvious short-term impact, but provides little to build on for the long term.
 - Stage two: Structured design techniques and data base-oriented languages come into use.
 - Stage three: Automated, integrated tools make the programmer's task much simpler. Limited direct user access removes some of the demand for programming services.
 - Stage four: Widespread direct user access, along with the use of select software building blocks, eases up to 75% of the previous programming workload.

3. RESULTS OF PRODUCTIVITY

- Performance characteristics: DP performance remains mediocre until the end of stage two, when performance begins to improve markedly.
 - Stage zero: Performance is correctly perceived as slow.
 - Stage one: Performance is perceived as slower, cost overruns are "normal," backlog of work is growing.
 - Stage two: Backlog is stabilized. "80% on time" becomes the performance target.
 - Stage three: On time, cost within budget is the rule, not the exception.
 - Stage four: Performance is assumed; it is no longer an issue.
- Productivity investment/payoff: The investment/payoff ratio will improve significantly as an organization moves to higher stages; but is not until the "efficiency" stage that the ratio is perceived to be positive, and it is only in stage four that the payoff becomes noticeably large.
 - This result is at least partially due to the long time required for an organization to appreciate and place a value on information as a factor of production.

D. SELF-ANALYSIS MATRIX DIAGNOSIS

- Data processing organizations should be able to rank themselves (and/or their constituent components) using the materials presented in this chapter. Exhibit IV-12 recapitulates the productivity stages and characteristics, and may be used directly as a checklist for ranking each column independently.

EXHIBIT IV-12

SYSTEM AND SOFTWARE PRODUCTIVITY SELF-ANALYSIS MATRIX

CHARACTER- ISTIC STAGE	EDP ORGANI- ZATION	APPLICATION TYPE	EDP MANAGEMENT DESCRIPTION	CONTROL MECHANISM	PERFORMANCE CHARAC- TERISTICS
ZERO: "CHAOS"	BUSINESS FUNC- TION "OWNED" BY USER	FUNCTIONAL COST REDUCTION, BATCH	PAWN; FOCUS ON OPERATIONS WORKING TOWARD CONTROL	VERBAL OR SIGNED PROJECT REQUEST	SLOW
ONE: "CONTROL"	CENTRALIZED BUSINESS FUNCTION	INTEGRATION OF OPERATING DIVISIONS	MANAGER; FOCUS ON COORDINATION, WORKING TOWARD QUALITY	BUDGET, SCHEDULE	COST OVERRUNS, SLOWER, BACKLOG GROWING
TWO: "QUALITY"	TECH SPECIALIST MATRIX PLUS BUSINESS FUNCTION, PROJECT BASIS	CROSS-FUNCTION- AL INTEGRATION, DATA BASE, ON-LINE	DIRECTOR; FOCUS ON PLANNING, WORKING TOWARD EFFICIENCY	FORMAL SDM, STEERING COMMITTEE	BACKLOG STABILIZED, 80% ON TIME
THREE: "EFFICIENCY"	MATRIX, TECH SPECIALIST AND BUSINESS SPECIALIZATION (USER MANAGER)	LARGE-SCALE DATA BASE, ON-LINE, SPECIALIZED AND STRATEGIC ORIENTATION	COMPANY OFFICER; CONCERNS BEYOND EDP; WORKING TOWARD VALUE	HIERARCHICAL STEERING COMMITTEE, LONG-RANGE PLAN, PRODUCTIVITY MEASUREMENTS	ON TIME, COST WITHIN BUDGET
FOUR: "VALUE"	DISTRIBUTED; CENTRAL GROUP DOES COMMON SYSTEMS, TECHNICAL ADVISORY	"WHAT IF" AND RE- TRIEVAL BY USER. CENTRAL GROUP DOES TECHNICAL SYSTEMS AND MAINTENANCE OF STRATEGIC DATA BASE RESOURCE	OPERATING MAN- AGEMENT OF COMPANY; FOCUS ON SURVIVAL, GROWTH AND PROFITABILITY OF COMPANY	PRODUCTIVITY INDEX, VALUE INDEX	ASSUMED, NOT AN ISSUE

EXHIBIT IV-12 (CONT.)
SYSTEM AND SOFTWARE PRODUCTIVITY SELF-ANALYSIS MATRIX

CHARACTER- ISTIC STAGE	QUALITY CONTROL	USER INVOLVEMENT	PROGRAMMING RESOURCES	PRODUCTIVITY TOOLS	INVESTMENT PAYOFF
ZERO: "CHAOS"	"IT WORKS"	REQUESTOR, DIRECT RELATION- SHIP ("OWNER")	BATCH/RJE	COBOL, FORTRAN, PL/1, BASIC, ASSEMBLER (!)	UNCONTROLLED COSTS
ONE: "CONTROL"	PRODUCTION VERSION VERSUS TEST VERSION	"LOCKED OUT"	RJE, SOME ON-LINE TERMINALS	TSO OR EQUIVALENT	BEGINNING TO CONTROL COSTS, CONSCIOUS IN- VESTMENT IN IM- PROVEMENT TECHNIQUES
TWO: "QUALITY"	SPECIFIC TEST PLAN, SEPARATE TEST PHASE AND QUALITY ASSURANCE GROUP	RELUCTANT PARTICIPANT, EARLY STAGES OF PROJECT	CLUSTERED OR SHARED TERMINALS, ASSURED DEVEL- OPMENT ACCESS	STRUCTURED DESIGN TECH- NIQUES, DATA BASE-ORIENTED LANGUAGES	CONTINUED INVESTMENT
THREE: "EFFICIENCY"	AUTOMATED TESTING, RELIABILITY A DESIGN ISSUE	PARTICIPANT THROUGHOUT PROCESS	TERMINAL PER PROGRAMMER AND ANALYST	AUTOMATED, INTEGRATED SET OF TOOLS, LIMIT- ED DIRECT USER ACCESS	EMERGING PAYOFF
FOUR: "VALUE"	QUALITY ASSURANCE A MAJOR DESIGN OBJECTIVE	ACTIVE LEADER, DOES SOME OF OWN SYSTEMS	SPECIALIZED, INTEGRATED DEVELOPMENT TERMINALS	WIDESPREAD DIRECT USER ACCESS, SELECT SOFTWARE BUILD- ING BLOCKS	SUBSTANTIAL PAYOFF

- It is sometimes difficult to obtain the desired objectivity when performing a self-analysis and self-rating. Consequently, it could be useful to obtain the assistance of an outside person who is knowledgeable about your organization (e.g., your auditing firm, a consultant who has worked with you, etc.)
- It will certainly be useful to obtain ratings from a variety of DP and non-DP managers, as well as from each level of personnel within the EDP organization, to serve as a basis for communicating problems and successes within the organization.
- The rating exercise should be performed at six- to twelve-month intervals to track the effectiveness of improvement measures instituted during the preceding period.
- In this case, it is essential to enlist the services of an objective third party in conducting the rating to eliminate any subconscious bias for or against the implemented "improvement."

E. SELF-ANALYSIS MATRIX PRESCRIPTION

- The importance of the characteristics defined in the columns of the self-analysis matrix is that they each present a progression of development toward higher productivity.
- Theoretically, each column could be pursued in isolation and result in some improved degree of productivity for the EDP organization.
- Practically, however, the columns are interrelated in the sense that organizational characteristics, for example, have a strong influence on the types and complexity of applications software, both those that are needed and those that can be attempted.

- Similarly, the degree of quality control that can be asserted depends to some extent on the available level of programming resources.
- Two factors must be kept in mind when using the self-analysis matrix to develop a productivity prescription for an individual organization:
 - By far the highest improvement in productivity comes from using information strategically. Efficiency improvements in productivity are of a much lower magnitude, although they are easier to see.
 - Strategic use of information depends on a balanced capability to produce and absorb a quantity and quality of information that is appropriate both to the organization and to the objective.
- These two factors, when applied to the self-analysis matrix, imply that each organization should attempt to progress smoothly and evenly through the five stages of productivity, and not attempt to develop great strengths in one area (e.g., programming resources) at the expense of a corresponding effort to develop the supporting areas (e.g., organization, quality control, user involvement, etc.)
- The initial use of the self-analysis matrix as a diagnostic instrument will show whether the organization has been able to achieve a balanced development, no matter what its stage of productivity.
 - Most of the characteristics will likely cluster along a single row of the matrix. To the extent that they do, this then becomes the target stage.
 - All other characteristics should be brought into alignment with the target stage before attempting to bring the entire organization to the next stage.
- If characteristics do not cluster around a single stage, laggard areas should receive extra attention, and areas that are in advance should be closely

inspected to see whether they are in fact helping to raise the organization to a new stage or whether they are only setting up unproductive tensions.

- At different stages of development, different characteristics will require emphasis to advance smoothly from one stage to the next. These varying priorities are shown in Exhibit IV-13.
 - In Stage zero ("Chaos"), the first priority must be to unify the EDP organization and establish controls.
 - After these have been achieved, the medium-priority items for this stage can be addressed; i.e., application types and programming resources.
 - The lowest-priority items contribute little at this stage, and may be left until last.
- To advance from "control" to "quality" still requires setting a high priority on EDP organization and control mechanisms, but re-establishing user involvement is equally high, since the user's support will be strongly required in the following stages.
- Making the transition to "efficiency" requires increased emphasis on productivity tools and programming resources to support quality goals.
 - EDP organization and application type have received sufficient attention to carry through this stage without a great deal of additional effort.
- The "value" stage requires that an ongoing high priority be placed on productivity tools, while applications are demanding a shift to development by the user, and therefore require renewed attention and support.

PRODUCTIVITY SELF-ANALYSIS MATRIX PRIORITY PROGRESSION

FACTOR STAGE	RELATIVE PRIORITY IN EFFORT TO REACH NEXT STAGE						
	EDP ORGANI- ZATION	APPLICA- TION TYPE	CONTROL MECHAN- ISM	QUALITY CONTROL	USER INVOLVE- MENT	PROGRAM- MING RESOURCES	PRODUC- TIVITY TOOLS
FROM 0 TO 1, "CHAOS" TO "CONTROL"	H	M	H	L	L	M	L
FROM 1 TO 2, "CONTROL" TO "QUALITY"	H	M	H	M	H	L	M
FROM 2 TO 3, "QUALITY" TO "EFFI- CIENCY"	L	L	M	H	M	H	H
FROM 3 TO 4, "EFFICIEN- CY" TO "VALUE"	L	H	L	M	M	L	H

DECISION RULE: FOR FACTORS MORE THAN ONE STAGE BEHIND, INCREASE PRIORITY ONE LEVEL FOR EACH LAGGING STAGE
NOTE: L = LOW, M = MEDIUM, H = HIGH

- Quality is, and will be, a continuing focus of development, but all of the other areas should by this time have been mastered.
- Since it is unlikely that any organization will find itself initially in a balanced state of development, a decision rule has been established that raises the priority within a column by one level for each stage that a given factor lags behind the "average" stage of the other factors.
 - For example, a company that is generally progressing from stage two to stage three shows many of the characteristics of both stages. However, the productivity tools in use may still be unstructured and batch-oriented, which are both stage-one characteristics. Consequently, the priority for addressing these factors should be raised from "L" to "M" or "H," depending on how far behind the condition is.
 - Factors that are a stage or two ahead should not be cut back, but neither should they receive additional attention until the companion factors are brought into line.

F. ACTIVITY-ORIENTED AUDIT OF PRODUCTIVITY

- In the short term, the EDP organization cannot usually take action on its own that will affect either the "determinant" factors described earlier (i.e., application types and EDP management role), or the corporation's overall attitude toward quality and organization-wide control mechanisms.
- However, there is much that the EDP organization can do to improve its own "instruments" of productivity:
 - The EDP organization itself (including the vital personnel function).

- Internal control and planning mechanisms.
 - User involvement.
 - Programming resources.
 - Productivity tools.
- Because of the great improvement opportunities available from the effective use of productivity instruments, a detailed audit of these activities is provided in Appendix E.
 - For each general area of the audit, specific approaches and tools that can be used have been listed, along with the relative "point score" for each approach.
 - Where the usefulness of a productivity approach varies, the point value associated with it will be different for different stages of development.
 - Please note that the numerical weights are useful working approximations that will permit progressive analysis over six- to twelve-month intervals. Refinements in the ratings are expected as organizations develop more experience with the audit.
 - As stressed earlier, no organization should try to use the entire "kit" of productivity tools at one time.
 - Many productivity strategies are equally applicable regardless of the development stage (e.g., strategies for training and morale improvement).
 - However, other strategies may be inappropriate at certain stages (e.g., a full-time, sophisticated planning function would be very important in

stages three and four, but would be counterproductive in the "chaos" stage).

- Rating materials are supplied in Appendix E for the following areas:
 - Organization and personnel:
 - . Organization.
 - . Objectives and motivations.
 - . Corporate policies.
 - . Recruiting and staffing.
 - . Turnover.
 - . Career development.
 - . Morale.
 - Internal control:
 - . Planning.
 - . Priorities and measurements.
 - . Control.
 - . Reporting accomplishments.
 - User relations:
 - . Involvement.

- . Responsibility.
- . Satisfaction levels.
- Programming support:
 - . Access to resources.
 - . Physical facilities.
- Productivity tools:
 - . Standards and methodology.
 - . Size and estimates.
 - . Analysis and design process.
 - . Programming process.
 - . Testing and certification process.
 - . Production.
 - . Enhancement and maintenance.
 - . Documentation.

V PROFILE OF PROCESS IMPROVEMENT AIDS

V PROFILE OF PROCESS IMPROVEMENT AIDS

A. RATIONALE FOR CATEGORIZATION

1. GENERAL

- Among the numerous ways in which productivity-related tools, aids and techniques can be categorized, the following classifications are meaningful and useful from a practical standpoint:
 - Relationship to a three-dimensional "problem space."
 - "Classical" tools versus "emerging" tools.
 - Applicability of given tools relative to system life cycle phase.
 - Impact of a given tool on a particular system life cycle phase.

2. THE THREE-DIMENSIONAL "PROBLEM SPACE"

- Software productivity problems come in many different varieties, each of which is amenable only to certain types of solutions.
- INPUT has defined a "Productivity Problem Space," which may be used to put in perspective the characteristics of desired solutions.

- The space, shown in Exhibit V-1, has three dimensions, defining the following characteristics:
 - Repetitive, simple systems versus complex, one-time systems provide the first dimension.
 - By INPUT's estimate, 70-80% of most organizations' workload falls into the repetitive, simple category; e.g., creating a new sales report.
 - Complex systems, such as implementing a communications network, are normally done much less frequently, but require a great deal of technical effort.
 - Short-term versus long-term solutions constitute the second dimension.
 - For this purpose, short-term is measured in days or weeks.
 - Long-term implies an effort extending over months or years.
 - Solutions that provide direct control over the problem, versus those that attack the problems indirectly, define the third dimension of the problem space.
- Exhibit V-2 shows various solutions discussed elsewhere in this report, plotted against the three dimensions of the problem space.
- To illustrate the meaning of Exhibit V-2, the first row should be read as follows:
 - To impact productivity problems with repetitive, simple systems, the following suggested solutions will provide short-term relief and will impact the problem directly:

EXHIBIT V-1

THE "PRODUCTIVITY PROBLEM SPACE"

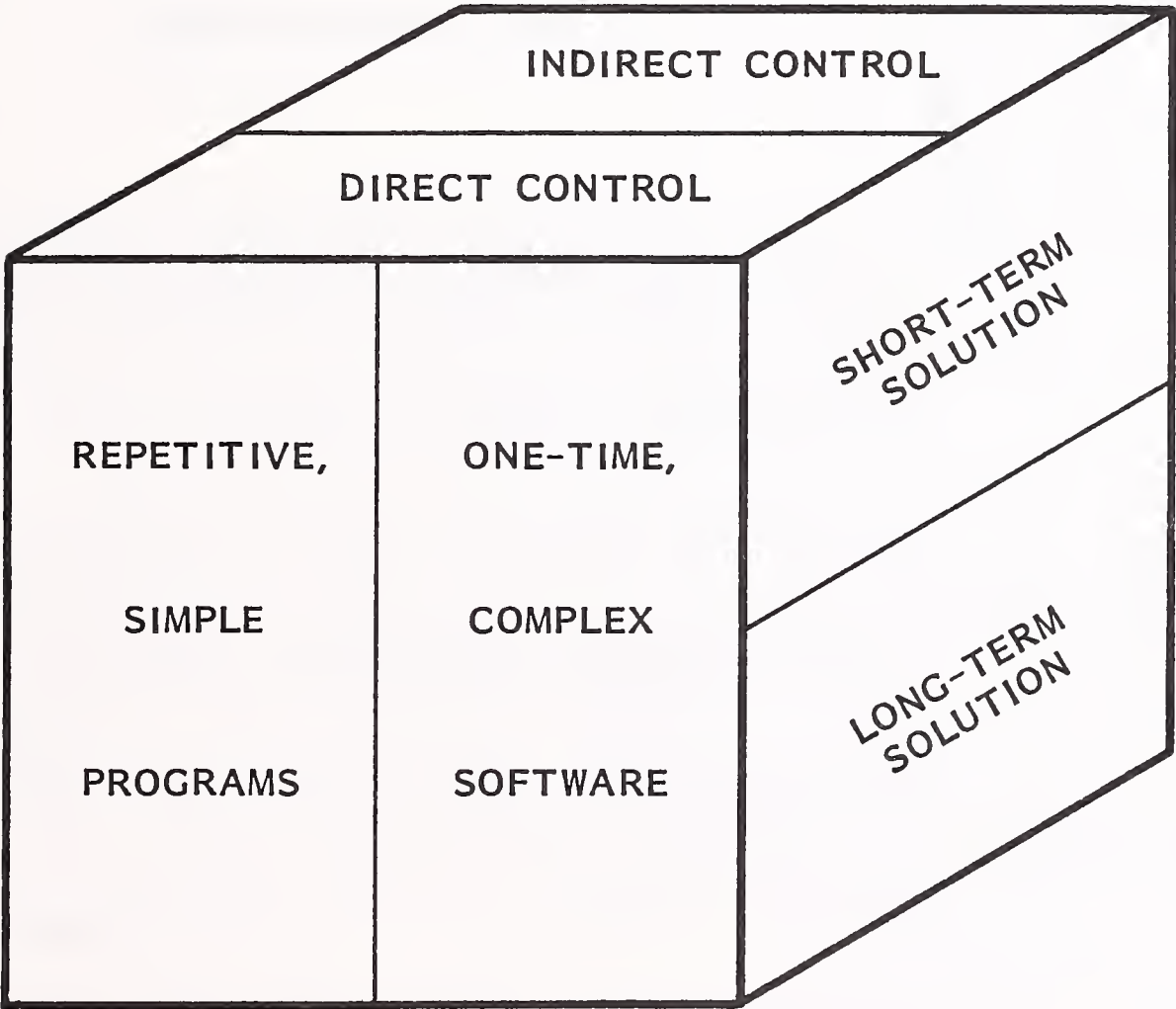


EXHIBIT V-2

SOLUTIONS RELATED TO PROBLEM SPACE

PROBLEM TYPE		SOLUTION IMPACT				APPLICABLE SOLUTIONS
REPETITIVE	ONE-TIME	DIRECT	INDIRECT	SHORT-TERM	LONG-TERM	
X		X		X		BATCHED MAINTENANCE, PROGRAMMER POOL, SCHEDULING, FACILITATE ACCESS TO RESOURCES
	X	X		X		BEST PEOPLE, ELIMINATE INTERRUPTIONS, FACILITATE ACCESS TO RESOURCES
X			X	X		USER PRIORITIZATION, MATRIX ORGANIZATION
	X		X	X		SIMULATIONS AND PROTOTYPES, FORMAL REQUIREMENTS LANGUAGE, MATRIX ORGANIZATION
X		X			X	USER LANGUAGE, TABLE-DRIVEN AND SKELETON PROGRAMS, MAINTENANCE TOOLS
	X	X			X	STEERING COMMITTEE, BUY INSTEAD OF BUILD
X			X		X	REUSABLE CODE, INTEGRATED TOOLS, EDUCATION IN IMPROVED TECHNIQUES
	X		X		X	ERROR AND DEVELOPMENT HISTORY LOGS AND ANALYSIS, REDUCE TURNOVER

- . Batched maintenance.
 - . Programmer pool.
 - . Scheduling.
 - . Facilitating access to resources.
- As another illustration, the seventh row reads as follows:
 - To impact productivity problems with repetitive, simple systems, the following suggested solutions will provide relief, but only in the long-term, and only through indirect action:
 - . Reusable code systems.
 - . Integrated tool systems.
 - . Education in improved techniques.
- In order to find possible solutions to repetitive system problems, look at the first, third, fifth and seventh rows. Direct approaches are listed in the first, second, fifth and sixth rows; and so on.
- The listing is not exhaustive, nor are the row entries mutually exclusive; but the solutions are representative, and serve as an index to other sections of this report.
- 3. "CLASSICAL" VERSUS "EMERGING"
- In a sense, every piece of systems software ever written is a productivity improvement tool, including:
 - Assemblers and macro-assemblers.

- Operating systems.
- High-level languages such as COBOL and FORTRAN.
- Anyone who finds this statement hard to accept need only compare the situation before and after these products came on the scene.
 - Before assemblers, people coded in binary or octal.
 - Before operating systems, every program had to carry its own I/O activities, down to the last channel command and status details.
 - Before high-level languages, all coding was done in assembly language or (previous to that) in machine language.
- However, as we enter the 1980s, certain software products have become such a firm part of the state of the art that they no longer need to be regarded as unique contributors to productivity. In addition to those already mentioned above, the following are included in this category:
 - Data base management systems (e.g., ADABAS, TOTAL, IMS).
 - Modern high-level languages (e.g., "C", Pascal).
 - Library systems (e.g., Panvalet, Librarian).
 - Preprocessors (e.g., Metacobol, S-Fortran).
 - Performance measurement aids (e.g., SMF, RMF, LOOK).
 - Word processing and text management systems (e.g., ATMS, STAIRS, SCRIPT).
 - Automatic flow charters (e.g., AUTOFLOW).

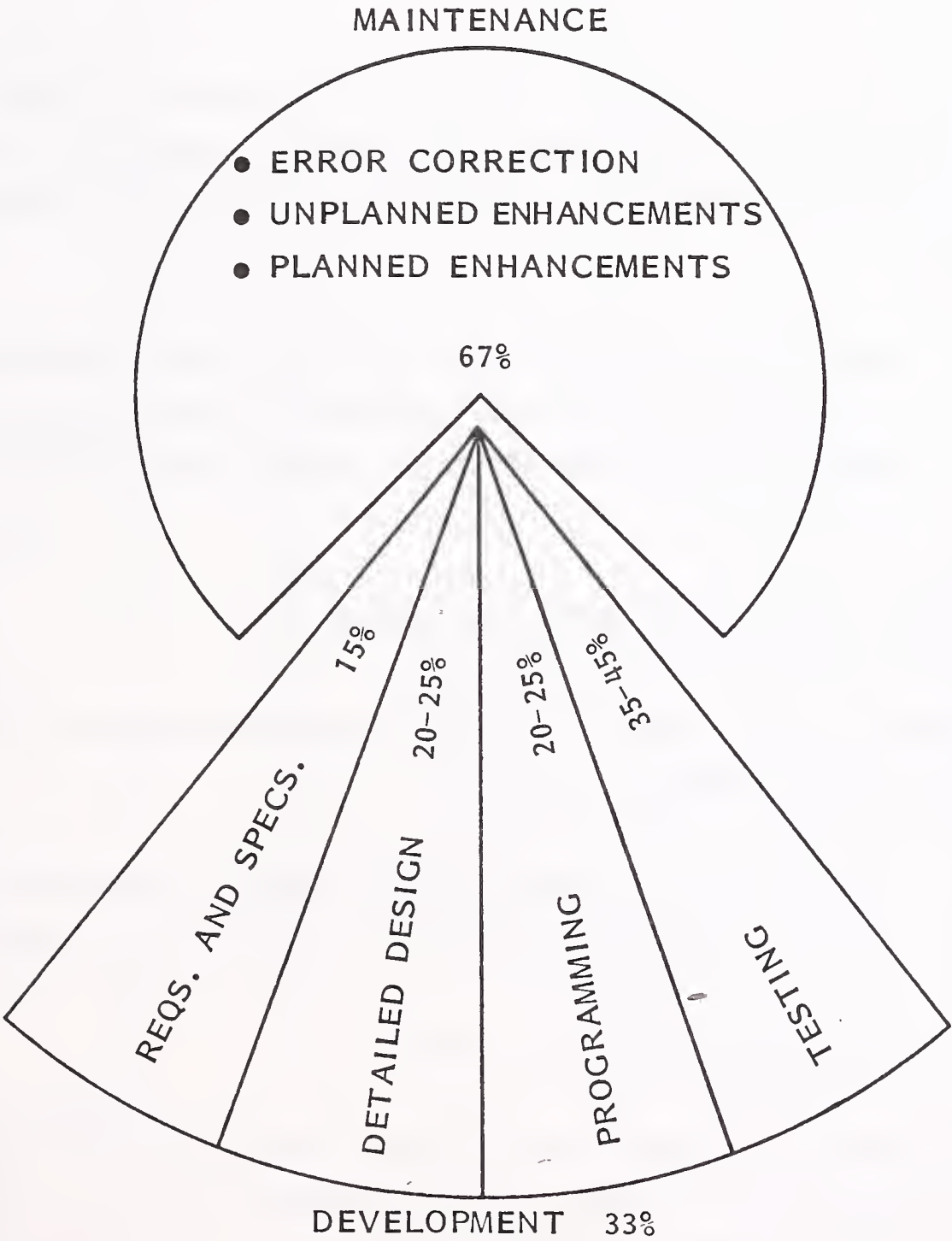
- On-line program development aids (e.g., TSO, CMS).
- Project control systems (e.g., Nichols, PAC II, PC/70).
- Although some of these "classical" productivity aids will be discussed in more detail later, the main purpose of this section is to introduce classes of "emerging-technology" productivity aids.
- These "emerging" tools and aids are generally relatively new, or else their use so far has been too limited to qualify them for the "classical" category.
- Specifically, the following classes of tools and aids are treated in this section as part of the emerging productivity-aid technology:
 - System design methodologies (e.g., PRIDE, SDM/70).
 - Requirements languages (e.g., PSL/PSA).
 - Organizational techniques (e.g., Chief Programmer Team, Matrix Organization).
 - Programmer's workbench (e.g., MAESTRO, UNIX/PWB).
 - Structured analysis (e.g., Jackson, SADT, Yourdon).
 - Menu-driven programming (e.g., DMS, ADF).
 - Reusable Code Systems (e.g., Raytheon's UPP).
 - Verification and validation systems.
 - Miscellaneous aids.

4. APPLICABILITY RELATIVE TO LIFE CYCLE PHASES

- The system life cycle may be divided into several subcycles as follows:
 - Development subcycle.
 - Requirements and specifications.
 - Detailed design.
 - Programming.
 - Testing.
 - Maintenance subcycle.
 - Error correction.
 - Unplanned enhancements.
 - Planned enhancements.
- Exhibit V-3 summarizes these cycles and subcycles, and gives INPUT's estimates of the relative costs of each subcycle. The important assumptions implicit in Exhibit V-3 are:
 - A typical system will last 7-10 years (including development).
 - The development subcycle is responsible for 33% of the life cycle costs (LCC).
 - The maintenance subcycle, which in this definition includes both planned and unplanned enhancements, is responsible for 67% of the LCC over the assumed 7-10 year life.

EXHIBIT V-3

LIFE CYCLE COSTS (LCC) FOR MAJOR PROJECTS
(≥ ONE PERSON YEAR)



ASSUMPTION: PRODUCT'S USEFUL LIFE IS 7 YEARS

- Different tools are appropriate for different portions of the life cycle. Exhibit V-4 is an attempt to show where in the system life cycle most of the activity employing a particular tool is concentrated.
- Note specifically that "applicability" in this sense may be quite separate and distinguished from "impact." Thus, a tool that may be applicable only during the requirements and specifications subphases may nevertheless have significant impact on, for instance, the maintenance phase much later in the project.
- As an illustration, the first row of Exhibit V-4 should be taken to mean that on-line development aids such as TSO and CMS are especially applicable during the coding and testing of a system, with some applicability to the maintenance phase.
 - Nevertheless, their impact on, and value to, the maintenance phase may exceed their value in coding and testing, especially to the organization whose operating characteristics demand low down-time for critical applications.

5. IMPACT RELATIVE TO LIFE CYCLE PHASES

- Exhibit V-5, shows the impact of the various tool categories on a specific life cycle phase or subphase.
 - "P" appearing in a given subphase means this particular class of tools can be expected to have its primary impact in that subphase.
 - "S" signifies a secondary impact.
- For example, structured analysis systems such as SADT or Warnier-Orr are expected to impact primarily the specifications phase, but they also have significant secondary impact on later phases.

EXHIBIT V-4 APPLICABILITY OF SELECTED TOOLS

TOOL \ LIFE-CYCLE PHASE	REQUIREMENTS	SPECIFICATIONS	DETAIL DESIGN	CODE	TEST	MAINTENANCE
ON-LINE SYSTEMS						
• CMS, TSO						
DBMS AND QUERY LANGUAGES						
• IMS, 204, ETC.						
SYSTEMS DESIGN METHODOLOGIES						
• PRIDE, SDM/70						
REQUIREMENTS LANGUAGES						
• PSL/PSA						
ORGANIZATIONAL TECHNIQUE						
• CHIEF PROGRAMMER TEAM						
PROGRAMMER'S WORKBENCH						
• MAESTRO, PWB/UNIX						
STRUCTURED ANALYSIS						
• JACKSON						
• SADT						
• STRUCTURED TABLEAU						
• WARNIER-ORR						
MENU-DRIVEN PROGRAM						
• DMS, TAPS						
VERIFICATION AND VALIDATION						
• STATIC AND DYNAMIC						
• STRUCTURED WALK-THROUGH						
MISCELLANEOUS						
• PDL						
• REUSABLE CODE						

----- POTENTIAL APPLICABILITY

EXHIBIT V-5
IMPACT OF SELECTED TOOLS

TOOL \ LIFE CYCLE PHASE	REQUIREMENTS	SPECIFICATIONS	DETAIL DESIGN	CODE	TEST	MAINTENANCE
ON-LINE SYSTEMS ● CMS, TSO				P	S	S
DBMS AND QUERY LANGUAGES ● IMS, 204 ETC.			P	S		
SYSTEM DESIGN METHODOLOGIES ● PRIDE, SDM/70	P	P	S	S	S	S
REQUIREMENTS LANGUAGES ● PSL/PSA	P	P	S	S	S	S
ORGANIZATIONAL TECHNIQUE ● CHIEF PROGRAMMER TEAM			P	S	S	
PROGRAMMER'S WORKBENCH ● MAESTRO, PWB/UNIX			S	P	S	S
STRUCTURED ANALYSIS ● JACKSON ● SADT ● STRUCTURED TABLEAU ● WARNIER-ORR	P	P P P	P S S S	S S S S	S S S S	S S S S
MENU-DRIVEN PROGRAM ● DMS, TAPS			S	P	S	S
VERIFICATION AND VALIDATION ● STATIC AND DYNAMIC ● STRUCTURED WALK-THROUGH			S	P	P S	S S
MISCELLANEOUS ● PDL ● REUSABLE CODE		S	S S	P P	S S	S S

NOTE: P = PRIMARY; S = SECONDARY

- This is due to such factors as improved readability of the resulting programs, which will have an impact on the maintenance phase, where programmers other than those who developed the program are usually called upon to understand the workings of that program.
- This particular case is an example of the differentiation between "applicability" and "impact."
- It should be apparent from this exhibit that there are currently no commercially available products or techniques that have their primary impact on maintenance.
 - Some "home-grown" tools fall into this category, however, and will be noted later in this chapter.

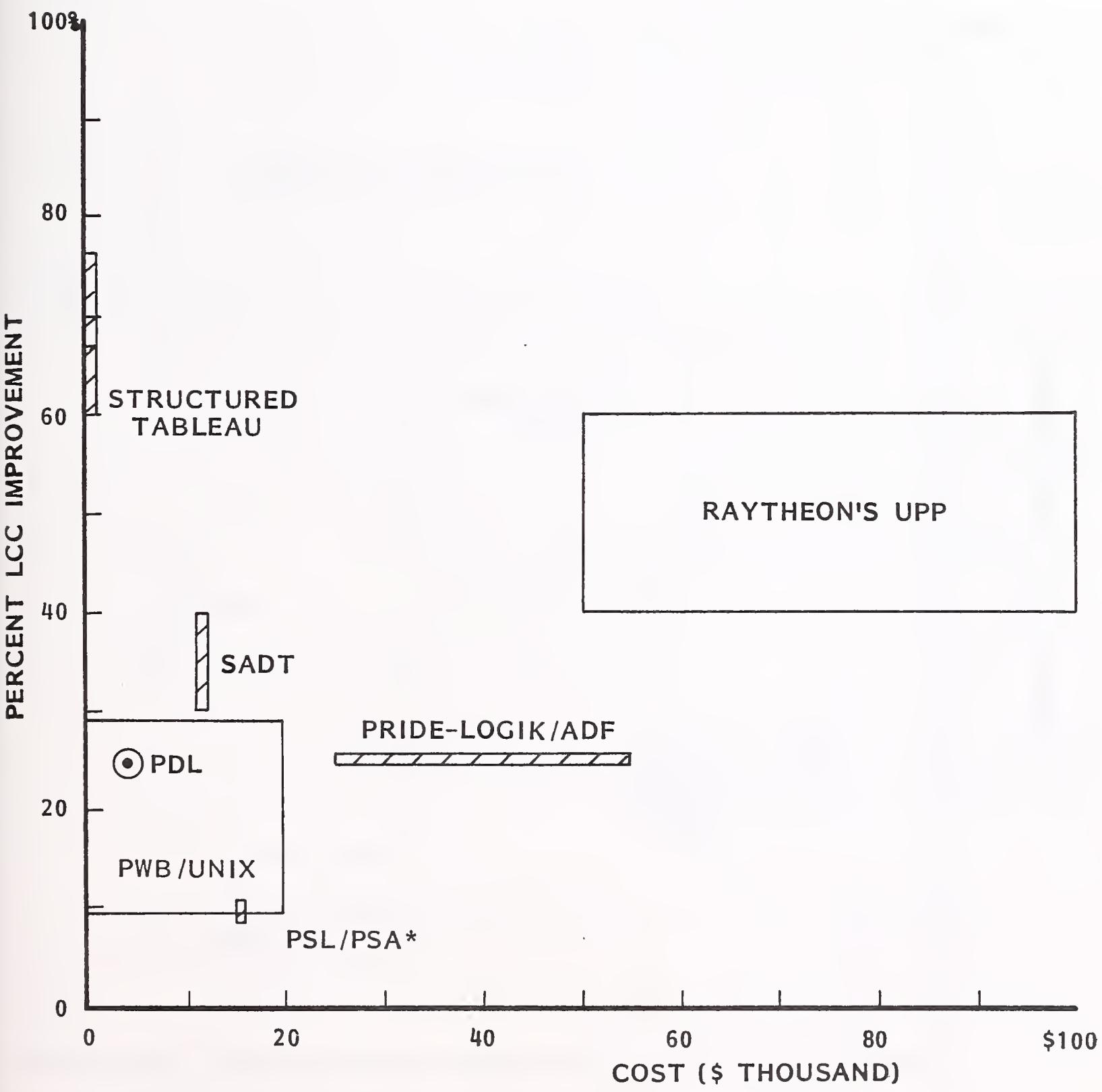
B. LIFE CYCLE COST IMPROVEMENTS VERSUS TOOL COSTS

- In the final analysis, the effectiveness of any proposed tool or technique must be measured by the degree to which it can reduce the total life cycle costs (LCC) of the project.
- The profitability of any proposed tool or technique can be measured by comparing the reductions in LCC to the costs associated with the introduction and use of the tool. Two of the most important and visible costs are:
 - The initial cost.
 - The learning-curve cost.
- Initial costs are, by and large, the costs of purchasing or licensing the tool or technique. Sometimes there are significant recurring costs, as in the case of PSL/PSA.

- Learning-curve costs are far more difficult to quantify. These are costs arising out of an initial decline in productivity, while people are "learning the ropes" of the new system and attempting to gain fluency in its use.
 - Many organizations interviewed felt that full benefits of a given tool or technique are not gained until the third successive project on which the tool is used.
- Establishing the LCC reductions associated with a given tool is a very difficult task.
 - There are few cases where valid "before-and-after" or "with-and-without" comparisons are available. Most organizations cannot afford the luxury of conducting controlled experiments in this area.
 - The various tools and techniques are in many cases promoted by "zealots" - not necessarily with a financial interest - who tend to accentuate the data supporting their favorite tool, while ignoring any evidence to the contrary.
- In the course of this research, INPUT has attempted to quantify the LCC reductions that can be expected from the effective use of a number of selected tools. How this was done is described in some detail later on in this section.
- Exhibits V-6 and V-7 summarize the results. They show the expected LCC reductions plotted against:
 - The initial cost (Exhibit V-6).
 - The learning curve (Exhibit V-7).

EXHIBIT V-6

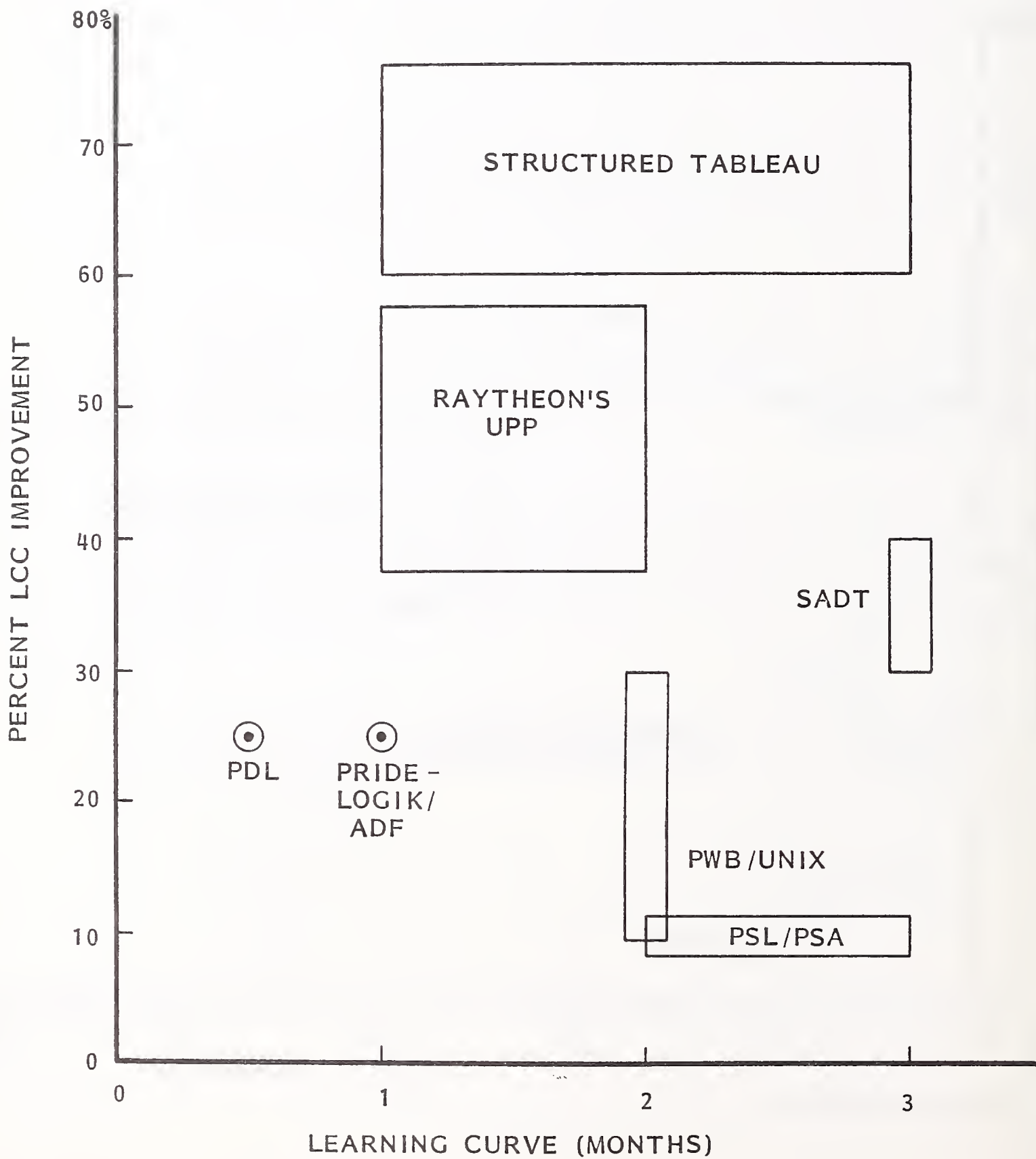
LCC IMPROVEMENT VERSUS INITIAL COST
FOR SELECTED TOOLS



*\$15,000 ANNUAL FEE

EXHIBIT V-7

LCC IMPROVEMENT VERSUS LEARNING CURVE FOR SELECTED TOOLS



- Because the learning curve costs are so hard to quantify, the X-axis shows the length of time (in months) that interviewees reported was required to become proficient in the use of each tool.
- Both the cost savings and the learning-curve costs are based on very small samples and contain a number of simplifying assumptions. It is essential to keep this in mind in evaluating Exhibits V-6 and V-7.
 - In particular, since there are absolutely no comparative data on the effects of various tools, these exhibits especially must NOT be interpreted as giving a hard, relative-value rating of tools.
 - Rather, these exhibits generally report how each tool performed relative to the case of no tools at all.
- It should also be clear that these tools are not at all competitive with each other, but could theoretically all be applied to a single project.
 - In practice, this has never been done, to the best of INPUT's knowledge.
 - Even if it were, tracing effects to causes would be virtually impossible.
- In the remainder of this section, the calculations of LCC improvements for selected tools are presented in some detail.
- In all the following calculations, the total life cycle costs are normalized to 1.0, with the following breakdown for subphases (based on the division shown earlier in Exhibit V-3):

- Requirements and specifications	0.05
- Detail design	0.07
- Coding	0.07
- Testing	0.12
- Total development cost	0.33
- Maintenance	0.67

- A caveat to prospective users of these techniques: there are no guarantees! INPUT has analyzed the figures reported, and proposes that they be considered as examples in the range of feasibility. Clearly, differences in skill and familiarity levels, as well as specific application characteristics and requirements, may produce different results for different users.
- Failure to include a specific technique within this section means only that data to support claims of productivity improvement for that technique were not sufficiently documented to be quoted.
 - A working assumption, that other techniques in the same category as those used for illustration will produce a similar magnitude of improvement, is warranted under the circumstances.
 - In each case where individual techniques are either the sole technique in the category, or are notably better than other members of the category, these facts have been stated.

I. STRUCTURED TABLEAU

- Good comparative data are available for at least four projects, as shown in Exhibit V-8.
 - Project "A" is a major on-line data base customer service system, initially done with classical techniques and based on CICS.
 - Project "B" is a complete rewrite of "A," using Franz's Structured Tableau design technique, and based on IMS.
 - Project "C" is a COBOL program that initially required 3,400 COBOL verbs and 78KB of memory (54KB in the Procedure Division). The program, developed by classical techniques, never became fully operational in 16 weeks of effort.

EXHIBIT V-8

LCC BASE DATA - STRUCTURED TABLEAU

MEASURED VARIABLES	PROJECT. DEVELOPMENT METHOD			
	<u>A</u> CLASS- ICAL	<u>B</u> STRUC- TURED TABLEAU	<u>C</u> CLASS- ICAL	<u>D</u> STRUC- TURED TABLEAU
MAINTENANCE STAFF	35	12	-	-
PERSON-HOURS TO COMPLETE	-	-	550	100
ELAPSED WEEKS TO COMPLETION	-	-	16	2
ESTIMATED MACHINE COST	-	-	\$4,000	\$500
RUN TIME (MINUTES)	-	-	90	90
MEMORY REQUIRED (KB)	-	-	78	40
- IN PROCESSING DIVISION ONLY	-	-	54	17
COBOL VERBS USED	-	-	3,400	750

- Project "D" is a complete rewrite of "C," using the Structured Tableau approach. It became operational after two weeks.
- From the data in the second and third rows, the LCC in the development phase appears to have improved five to eight times over the base case, yielding a development phase LCC of 0.04 to 0.07.
- The data in the first row indicate a maintenance improvement factor of two to three, yielding a maintenance phase LCC of 0.20 to 0.33.
- The total LCC using the Structured Tableau technique is therefore judged to range from 0.24 to 0.40, which translates to an LCC improvement of 60% to 76%.

2. PDL

- Exhibit V-9 summarizes some empirical data reported by Caine, Farber & Gordon, Inc., the developers of PDL.
 - Project "A" is a major component of a seismic data processing system for oil exploration. This project was developed using "plain vanilla" structured programming techniques.
 - Project "B" is a system for the automatic structuring of FORTRAN programs. It was developed using PDL and structured programming.
- In addition to the data in Exhibit V-9, Caine, Farber & Gordon (CFG) also reported that, in a number of complex projects, they experienced 60-65 lines of code (L.O.C.) per person day (over the entire development cycle), and computer utilization of 0.25 CPU-hours per 1,000 lines of finished code.
 - CFG reported that the L.O.C. figures are about 1.5 times better than their own best efforts using "plain vanilla" structured programming techniques alone. They estimate that the line of code rate is 4-6 times

EXHIBIT V-9

LCC BASE DATA - PDL

MEASURED VARIABLES	PROJECT. DEVELOPMENT METHOD	
	A "PLAIN VANILLA" STRUCTURED PROGRAMMING	B STRUCTURED PROGRAMMING WITH PDL
LINES OF CODE/PERSON DAY*	40	65
CPU-HOURS/L.O.C.**	0.90	0.16
LANGUAGE	PL/1 DIALECT	PL/1
PROGRAM SIZE IN L.O.C.	32,000	27,000
TEAM SIZE	3-6	3-5
ELAPSED MONTHS	9	6

*CALCULATED OVER ENTIRE DEVELOPMENT CYCLE

**370/158

better than the average industrial experience using "classical" techniques.

- According to CFG, the computer usage figures, according to C-F-G, are four times smaller than their own experience using structured programming alone, and ten times smaller than the classical industrial average.
- Using the production (L.O.C.) figures given above and in the first line of Exhibit V-9, INPUT estimates that the use of PDL can lead to a 4:1 improvement (75% reduction) in the development phase.
 - No hard data exist to show any significant impact on the maintenance phase. INPUT therefore assumes that the use of PDL will not result in any significant cost reductions in that phase.
- Based on these calculations, the total LCC improvement due to the use of PDL is estimated at 25%, as follows:

- Development cycle	0.08 (25% improvement)
- Maintenance cycle	0.67 (no change)
- Total LCC	0.75, (25% improvement)

3. SADT

- Two sources reported a 20% improvement in the development phase.
- One source reported a total LCC reduction of 30% to 40%.
- Another source reported a reduction in the number of "bugs" found during integration and testing by factors of two to 10.
- Taking the figure of 20% for the development phase as a base, the LCC reductions in the maintenance phase are estimated at 35% to 50%, which would yield an overall 30-40% LCC reduction.

- Thus INPUT estimates that the LCC improvements due to the use of SADT are as follows:

-	Development phase	0.26 (20% improvement)
-	Maintenance phase	0.34 - 0.44 (35%-50% improvement)
-	Total LCC	0.60 - 0.70 (30%-40% improvement)

4. PSL/PSA

- INPUT estimates that the use of this tool should result in the following changes in the LCC for various subphases:

-	Requirements and specifications	0.06 (20% penalty)
-	Detail design	0.06 (14% improvement)
-	Coding	0.07 (no change)
-	Testing	0.11 (20% improvement)
-	Total development Phase LCC	0.30 (9% improvement)
-	Maintenance phase	0.60 (10% improvement)
-	Total LCC	0.90 (10% improvement)

5. REUSABLE CODE

- The developers of the Raytheon reusable code system reported that they had analyzed a large number of systems and concluded that 40% to 60% of the functions being coded in any given system had been previously coded in another system.
- Taking this to mean that an LCC improvement of 40% to 60% is possible in all phases of the life cycle, excluding possibly the requirements and specifications phase, INPUT estimates that total LCC improvements from 38% to 57% are possible with reusable code.

6. PWB/UNIX

- Data supplied by one source INPUT interviewed indicate the following improvements:
 - Thirty percent due to the use of the Screen Editor.
 - Ten percent due to the specific PWB features (MAKE, SCCS).
 - Twenty-five percent due to the ability to incorporate existing code segments into new programs.
- INPUT estimates that these figures are roughly equivalent to an improvement of 10% to 30% in all phases, except possibly the requirements and specifications subphase.
 - This then translates into a total LCC improvement of 9% to 29%.

7. PRIDE/Logik

- INPUT estimates the following LCC changes due to the use of a system design methodology such as PRIDE/Logik:

-	Requirements and specifications	0.06 (25% penalty)
-	Detail design	0.09 (25% penalty)
-	Coding	0.06 (20% improvement, assuming GENASYS)
-	Testing	0.10 (30% improvement)
-	Maintenance	0.44 (35% improvement)
-	Total LCC	0.75 (25% improvement)

C. FEATURE ANALYSIS OF SELECTED TOOLS AND TECHNIQUES

I. ON-LINE PROGRAM DEVELOPMENT AIDS

- For at least ten years now, there has been an almost universal belief that on-line access to program development tools results in greater programming productivity.
 - In INPUT's mail survey, the largest group of respondents named on-line interactive program development as the single most successful factor for productivity improvement.
- This belief is supported primarily by logic.
 - It is reasonable to assume that the periods of waiting for batch outputs of trial runs are largely unproductive.
 - Even when the programmer attempts to multiplex the time by working on program B while waiting for program A's output, the mental "gear shifting" from one line of thought to another probably degrades productivity.
- Experimental evidence comparing on-line to batch environment has always been scarce, and in recent years efforts to obtain such data appear to have ceased altogether, because the trend to interactive operations has simply become overwhelming.
 - Results were mixed in the few studies that were performed, and in general no clear productivity advantage to either method was discovered (cf. Sackman).

- However, these studies were done when teleprinters were the principal user terminals; the effect of full-screen support CRT was not reflected in these studies.
- There was clear agreement that on-line environments result in substantially greater computer resource requirements (cf. Sackman).
- There is a nagging suspicion that excessive on-line access can be detrimental to productivity by encouraging sloppy work. This suspicion was expressed by a number of people INPUT interviewed. It has also been reported in other industry literature.
 - However, it can also be reasonably argued that this fault is not inherent in the interactive environment and can be avoided by establishing discipline through appropriate management practices.
- The elements of the interactive development environment are by now common to many varieties of hardware and to many software monitors. They include:
 - A mechanism for controlling the communications lines to a number of terminals.
 - An editor to allow terminal operators to enter, modify and examine text, especially source code and listings.
 - A command language to enable terminal users to invoke assembly, compilation, link editing and execution of programs, often with dynamic tracing capabilities and symbolic debugging features.
 - A "HELP" menu to guide the uninitiated in the details of the interactive system.

- When these elements are reentrant, the efficiency of the system is greatly enhanced by the ability to service multiple editing operations with a single copy of a reentrant editor.
- TSO and CMS are the two best-known on-line systems from IBM.
 - TSO ("Time Share Option") is designed to reconcile the batch orientation of IBM operating systems, especially MVT, SVS and MVS, with the need to service interactive program development and testing. TSO combines the functions of a communications monitor, controlling a large number of users' terminals, with such services as interactive data set creation, text editing, compilation and test runs. To the operating system, TSO appears to be just another "job."
 - CMS (Conversational Monitor System) is the "native" operating system of VM/370, a system designed to permit execution of many different operating systems on a single CPU. CMS services a single-terminal user with facilities similar to TSO. By running multiple "CMS machines," a VM-controlled CPU can create a multiterminal interactive environment similar to TSO.
- More recently developed operating systems from other vendors do not need artifices like TSO or CMS, because they are inherently designed to support an interactive environment.
 - The TOPS10 and TOPS20 operating systems for the DEC PDP-10 and 20 (now called DEC systems 10 and 20) are examples of an operating system designed to support both batch and interactive workloads.
 - The operating system for the Wang 2200 and VS computer lines are entirely interactive-oriented, and assume that each user has a CRT workstation. Batch workload is handled by treating the batch input and output as if they come from, or are destined to, a dummy or simulated interactive terminal.

- The UNIX operating system for the DEC PDP-11 computers is another example of a completely interactive-oriented system.
- Other on-line access systems for IBM mainframes include:
 - VSPC (Virtual Storage Personal Computing), an IBM program product for DOS/VS, OS/VS1 and MVS. As the name implies, VSPC is oriented toward providing an interactive environment for solving personal computing needs, rather than toward developing large, production-type applications. However, it has been applied to these situations by some organizations.
 - ETSS (Entry Time Sharing), an FDP specialized for DOS/VS.
 - ICCF (Interactive Computing and Control Facility), an IBM program product version of the ETSS software designed specifically for DOS/VSE with AF.
 - ROSCOE, a conversational, remote-job-entry system offered by ADR as an alternative to TSO, running under OS, OS/VS1, OS/VS2.
 - WYLBUR, another conversational RJE system originally developed at Stanford University, offered in various enhanced-supported versions by a number of commercial software houses, and run under OS and OS/VS.
 - TONE, another TSO substitute, marketed by Tone Software Corp., with versions available for OS/VS1-SVS and for MVS.

2. PROGRAMMER'S WORKBENCH

- The term "Programmer's Workbench," abbreviated PWB, is, strictly speaking, the name of a specific product from Western Electric.

- However, the concept of PWB is more general. Systems with similar intent from other manufacturers include:
 - Maestro from Softlab GmbH/Maestro Systems Inc.
 - Programmer's Workstation (PWS) from Four Phase.
- The intent of all of these systems is to create an interactive working environment in which the programmer has ready access to a uniform set of tools designed to facilitate program development, testing, modification and documentation.
- In addition, all three systems have the capability of operating in "front end" mode. That is, the PWB system can run on a small CPU attached to a larger "host" machine, thereby "offloading" from the host the overhead associated with servicing the program development activities, and freeing the host to work exclusively on production systems.
 - Although all three systems can be used similarly with IBM 360/370 hosts, Maestro is especially tailored for this environment and is being marketed as a TSO replacement.
 - When the system operates in this mode, compilations and executions take place in the host.
- PWB, as described in Exhibit V-10, is not a standalone system, but an adjunct to the UNIX operating system (Release 6).
 - PWB/UNIX, developed by Bell Labs, is marketed by Western Electric on an "as-is" basis, with no support and relatively minimal documentation.
 - Several firms, such as Interactive Systems Corp. of Los Angeles, are licensed by WE to resell PWB/UNIX; such resellers typically add features, documentation and support.

EXHIBIT V-10

TOOL PROFILE: PWB /UNIX

NAME

PWB /UNIX

TYPE AND APPLICABILITY

Programmer's workbench; applies to specifications, detail design, coding, test and maintenance phases.

MAIN BENEFITS

- Ability to reconstruct status of software or system at current and each previous stage
- Ability to easily incorporate existing code into new designs
- Aids communications via electronic mail
- Standard benefits of on-line program development and testing (full-screen editor, etc.).

ESTIMATED LCC IMPROVEMENT

Requirements and specifications - no change

Detail design

Coding

Test

Maintenance

} - 10-30% (INPUT's estimate)

Total LCC improvement: 9-29%

COSTS

From none (educational usage) to \$20,000 (commercial resellers)

TRAINING

One day with previous UNIX experience; two months' self-study of UNIX and PWB with a background in another computer system.

LEARNING CURVE

Good people can become proficient in use of PWB /UNIX in about two weeks after training

DISADVANTAGES

- Poor user documentation, even from resellers
- No on-line prompting
- No ongoing support from WE

SUPPLIER

Mr. Al Arms

Western Electric

POB 25000

Greensboro, NC 27420

(919) 697-2861

- One of the interesting aspects of PWB/UNIX is the fierce loyalty that it seems to generate among its users, most of whom are from the engineering or academic communities. One organization interviewed reported that the programming staff in one division felt so strongly about the system that efforts to replace it with another, corporate-wide system were shelved.
- PWB cannot be separated from UNIX. The entire PWB/UNIX system must be regarded as a whole in evaluating its impact on programmers' productivity. Many of the features that make the system so attractive to its fans are inherent in the interactive-oriented UNIX in all its versions. However, the PWB component is integrated into one particular version of UNIX, Release 6.
- The PWB/UNIX offers the following features:
 - Hierarchical, tree-structured file/directory system.
 - An easy-to-use command system (the "shell"), which is extensible by the user. Some programming tasks can actually be coded entirely in shell commands, a facility that is often used as "disposable" code for developing "first-cut" applications quickly.
 - A powerful text-processing system for the creation and modification of general text (e.g., program documentation) as well as source code.
 - The Source Code Control System (SCCS), a set of commands to facilitate the storage, updating and retrieval of source code and/or document files by date or version number. The exact status of the code or text at any particular point in the history of the project can be reconstructed. Identifying information can be automatically inserted into source code modules to establish the version and release of the module, given only the corresponding load module or merely a memory dump.

- Remote job entry (RJE) facility for the submission of jobs and retrieval of output to and from IBM System/370 hosts via HASP, ASP or JES2.
- The "C" programming language, which supports structured programming constructs (high level), but also provides for bit manipulation (low level).
- The disadvantage cited most often by the PWB/UNIX users interviewed was that the documentation, even from resellers, was sketchy and was not oriented to first-time users. In fact, many of the PWB/Unix supporters are from either the engineering or academic communities, and are not at all first-time users. Other disadvantages cited were:
 - Lack of on-line prompting.
 - Lack of support from WE.
- Maestro consists of software developed in Germany by Softlab GmbH, and hardware from Four Phase Systems. It is a packaged system intended to operate as a "front end" to an IBM 370 CPU, or any other facility that recognizes the 3270 protocol. Compilations and executions take place in the host.
 - Maestro is supplied in the U.S. by Maestro Systems Inc., 3 Embarcadero Center, Suite 2470, San Francisco, CA 94111.
- The Maestro system contains an integrated set of program preparation tools, including:
 - Support of "Structograms" (actually Nassi-Schneiderman charts).
 - Support of HIPO charts and decision tables.

- Text entry, modification and display capabilities, enhanced by syntax prompting for COBOL, FORTRAN and PL/I.
 - Shorthand capability via programmer-defined, single-keystroke abbreviations of any recurring character string.
 - Skeletal programs, programmer notes, system documentation, test results, work in progress, previous versions, etc., can all be tied together via programmer-defined "bookmarks" to permit immediate screen reference back and forth between up to twelve separate files of information.
 - Other features to be announced early in 1981.
- Current users of Maestro report that it is more oriented toward the commercial or business systems environment than is PWB. They also report savings based on the ability to support up to 24 programmers for a direct cost of approximately \$300,000 (minimum 10 programmers at \$200,000), compared to the cost of supporting them on the host via TSO.

3. STRUCTURED PROGRAMMING/DESIGN/ANALYSIS

- In 1966, Corrado Bohm and Giuseppe Jacopini published a paper in which they proved that any program that can be represented by a conventional flow chart can be constructed exclusively from the three elements or constructs illustrated in Exhibit V-11:
- Sequence.
 - Iteration.
 - Selection.

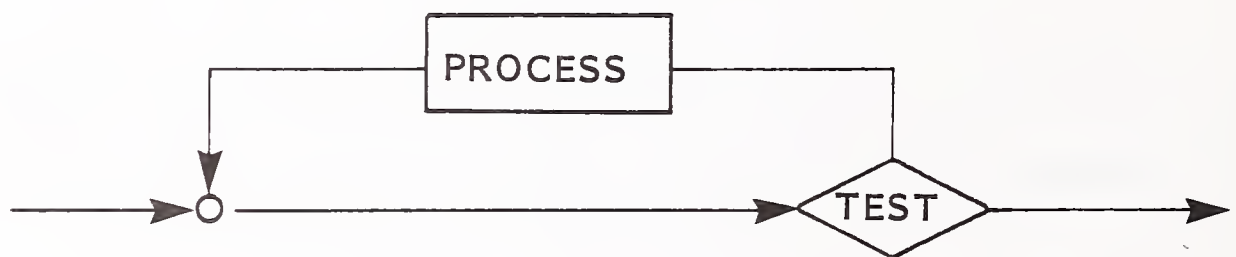
EXHIBIT V-11

STRUCTURED PROGRAMMING CONSTRUCTS

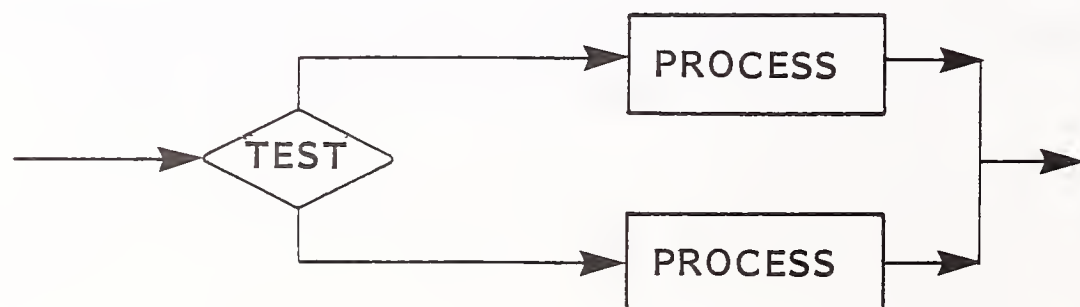
- SEQUENCE



- ITERATION



- SELECTION



- This assertion has become known as the "Structure Theorem." It lies at the heart of an entire programming and design discipline, called "structured programming," which intentionally restricts all programs to the use of these three basic constructs.
- In the late 1960s and early 1970s, Edsger W. Dijkstra of Holland, Niklaus Wirth of Switzerland, and Harlan Mills of the United States developed the ideas of structured programming into consistent, more general programming methodologies. Later efforts by Douglas Ross, Larry Constantine, Edward Yourdon, Michael Jackson, Tom DeMarco, Chris Gane, Trish Sarson, Kenneth Orr, Jean-Dominique Warnier and others extended the "structure" concepts into the new disciplines of "structured design" and "structured analysis."
- Because structured programs require no "GO TO" branching logic, the technique was initially known as "GO-TO-less" programming. It is now widely understood that the absence of GO TOs is a relatively minor consequence of the much more encompassing principles at the root of structured programming and design.
- There was in the early 1970s a general euphoria about the future of structured programming and design. Brave predictions proclaimed that the nature of programming would quickly change from that of "black magic" to a firmly regulated engineering discipline.
 - It is now clear that these predictions were far too optimistic, and that structured techniques, while valuable, are only a part of the arsenal of tools and strategies that must be employed to attack the productivity problem in programming.
- By far the major benefit of restricting programs to only three elementary constructs is stylistic. Programs written in this way are easy to read, because they have an inherent "top-down" property:

- Unlike programs with unrestricted branching logic, structured programs can be read sequentially, from the top down, without ever having to "jump around" to follow GO TO logic.
- Nevertheless, this matter of common style is at the heart of the productivity improvement directly attributable to structured programming.
 - Enforcing a common style facilitates communication between team members during the development phase.
 - By making programs more understandable to people other than the original designers and coders, it is reasonable to assume that the technique also aids in the maintenance phase.
- Other benefits of structured programming include the following:
 - Because structured programs are inherently sequential or "top-down," they are especially appropriate for hierarchical coding disciplines and the implementation of hierarchical, top-down designs.
 - Because structured programs have just three basic constructs, they can be expressed in terms of mathematical function theory. This makes it hypothetically possible to construct rigorous "correctness proofs" for such programs. In practice, such proofs are rarely useful because they require more effort than empirical debugging.
- The main disadvantage of pure structured programming (i.e., free of other, essentially independent techniques such as hierarchical design) is that it generally leads to less efficient programs in terms of both required storage and execution time.
 - With the declining cost of hardware in general, and memory in particular, this is a far less serious objection today than it was in the early 1970s.

- Perhaps more to the point is the objection that strict adherence to the structured programming rules limits programmers' creativity, which may be the reason why many programmers resist the technique.
 - Nevertheless, many people would argue that a simple-minded, straightforward solution is preferable to a brilliant, convoluted solution.
- It is difficult to enforce the structured discipline with the major programming languages today, which provide much more sophisticated control and branching structures than those allowed by pure structured programming; or conversely, only awkwardly permit the limited "IF-THEN-ELSE," "DO-WHILE" and "CASE" structures.
 - A partial solution is to provide a "structured preprocessor," or macro facility, to support the structured construct exclusively, or in addition to the native facilities. Examples are METACOBOL from ADR, S-FORTRAN from CFG, and others.
 - Another solution is to standardize on a less rigorous, but still constrained, set of alternatives, such as "Forward GO TOs only, and only within the module," or "Nested IFs no more than three levels deep."
 - Perhaps the nearest approach to "structured" language is PASCAL, originally developed by Niklaus Wirth at the ETH University in Switzerland. The language has richer branching logic than the minimum allowed, but still avoids the GO TO construct entirely.
- A large number of "structured methodologies" for requirements analysis, system specification, overall design and even detail (program) design have been developed over the past ten years or so. They include, among others:
 - Constantine/Yourdon/DeMarco Structured Design.
 - Jackson's Program Design Methodology.

- Warnier-Orr's Structured Design.
 - Gane & Sarson's Structured Design.
 - Ross's SADT.
 - IBM's HIPO.
 - Franz's Structured Tableau.
- While these systems differ in many respects, they generally share several characteristics:
 - They are related to structured programming in some way, typically facilitating the use of the constructs of sequence, iteration and selection by the manner in which they define the output of the analysis and design tasks.
 - They all employ some graphical system of representation as the main vehicle of the design, documentation and review (human communications).
 - They all emphasize top-down, hierarchical design concepts.
 - They are all concerned with the creation of logical models of systems, answering the questions of "why" and "what," rather than "how."
 - With the exception of the last three, all others take the flow and/or structure of data as a point of departure, with procedure design following directly from the data design (e.g., Jackson), or being complementary to it (e.g., Warnier-Orr, Gane & Sarson, Yourdon, et al.).

- SADT is an exception in that it gives equal weight to the design of data structures and process structures.
- HIPO is an exception in that it concentrates entirely on the design of procedures.
- Structured Tableau is also a procedure-oriented design technique, relying on decision tables as the primary graphic tool.
- These "structured methodologies" are in essence intellectual disciplines or frameworks for the analysis and design tasks. They can be used in conjunction with such other tools and strategies as:
 - System development methodologies; e.g., SDM/70, Pride/Logik, etc.
 - Organizational techniques; e.g., Chief Programmer Team, Egoless Programming.
 - Review/interaction techniques; e.g., structured walk-throughs.
 - Data dictionaries.
 - Code generators, pseudo code, PDL, etc.
 - Requirements languages; e.g., PSL/PSA.
 - Other tools and aids.
- While each structured methodology claims a particular set of benefits, these benefits can be generalized for the group of methodologies as a whole.

- Concentration on the logical requirements and structure of systems in the design phase, and delaying the "how" consideration to later stages, results in more robust designs that reduce the need to make major about-faces later on in the life of the system.
 - Use of graphical descriptions, with their meanings more precise than natural language, promotes clarity in communications between the end users and developers, among the development team itself, and between the developers and the maintenance personnel.
 - Use of top-down, hierarchical design results in gradual exposition of detail, which makes the design intellectually manageable.
- By the same token, these methodologies also share some common problems.
 - The most obvious problem is that substantially more time is spent in the analysis, specification and design phases, creating in some cases user apprehension and management impatience.
 - Training is another significant problem. While the mechanical details of most of these methodologies are relatively easily mastered, their effective utilization depends on a radical change in the way people think about systems. This change requires repeated experience and exposure to the basic principles of the structured methodology. Experienced personnel have frequently resisted making such a radical change.
 - Finally, these methodologies are, in effect, imprecise intellectual frameworks that leave a great deal to the ingenuity, insight and resourcefulness of the designer. That is, there is no guarantee that, given the same problem, any two designers using the same methodology would follow essentially the same path to a solution, or even that they would reach the same solution.

4. DATA-DRIVEN STRUCTURED METHODOLOGIES

- Some of the more popular structured design methodologies today share the common characteristic that they assume data design precedes the design of procedures. They assume, in effect, that procedures may be designed directly from the data design. Such methodologies may be characterized as "data-driven." They include:
 - Yourdon, DeMarco, V. Weinberg.
 - Jackson.
 - Gane and Sarson.
 - Warnier-Orr.
- Among these, the Jackson Program Design Methodology is somewhat unique in that it:
 - Specifically addresses detailed program design, rather than total system design.
 - Uses simple graphics and does not rely on the Data Flow Diagram.
- A group of methodologies that can be characterized as the "Yourdon school" are described in books published by people who are, or once were, associated with the Edward Yourdon firm. These include:
 - Yourdon, DeMarco, V. Weinberg.
 - Gane and Sarson.
- All the "Yourdon School" methodologies employ the Data Flow Diagram (DFD) as one of the major design communication and documentation tools.

- The DFD, also called a "Bubble Chart," is a network of circles (bubbles) connected by arrows. The bubbles generally represent processes or activities, while the arrows show the incoming and outgoing data.
- In addition to the DFD, these methodologies also employ additional tools or documentation, including:
 - Data dictionary, containing detailed definitions of the interfaces (data flows) declared on the DFD.
 - "Minispecs," a definition of the processes shown on the lowest level of the DFD set, in terms of "structured English," decision tables, trees and other non-linguistic artifices. Minispecs are the input to the detail design stage.
 - The Data Immediate Access Diagram (DIAD) and the Data Store added by Gane & Sarson.
- In the Warnier-Orr methodology, the graphic language and documentation aids are different. They are:
 - The Warnier-Orr diagrams.
 - Entity Diagrams.
 - Problem Matrices.
- In the Warnier-Orr methodology, the design starts by identifying the desired outputs of the new system.
 - This is followed by the data design, both logical and physical.
 - Finally, the procedures that manipulate the data are designed.

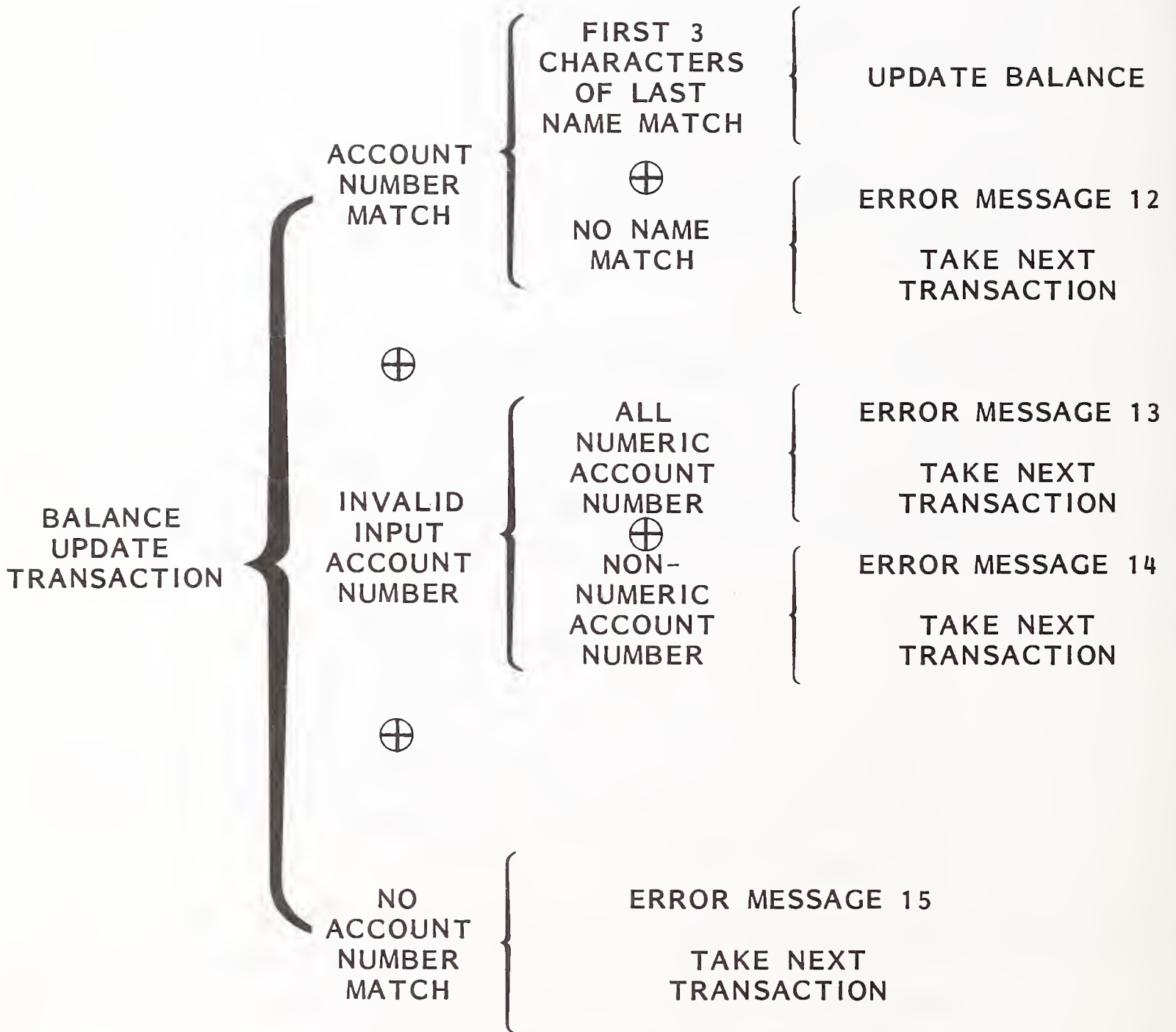
- The Warnier-Orr diagrams consist of a sequence of successive, left-to-right, braces, with notations for identifying the constructs for sequence, alternation, repetition and concurrence, as shown in Exhibit V-12.
 - Subordinate levels are placed to the right of superior levels.
 - Each brace is a level.
- The entity diagram is a form of the DFD, with bubbles representing entities and arrows representing data flows or lines of communication.
- Problem matrices are used to document such problem/solution aspects as the symptoms of the problem, the objectives of the solution, the cause of the problem, etc.

5. SADT (STRUCTURED ANALYSIS DESIGN TOOL)

- SADT, commercially available from SofTech in Waltham, MA, is interesting both because of the characteristics it shares with other structured design methodologies and because of the characteristics that set it apart.
- SADT shares with the Yourdon School the following concerns:
 - Viewing the analysis activity as logical model building, where the models are intended to answer questions of purpose and objectives ("why") and of "what" the system is supposed to do, rather than "how" to implement the system.
 - Employing top-down decomposition, whereby complex systems are described in successive layers that expose greater and greater detail, until the final level of desired detail is reached.
 - This makes complex models of systems intellectually manageable.

EXHIBIT V-12

WARNIER-ORR DIAGRAMS



- Unlike the Yourdon School, Jackson and Warnier-Orr, SADT gives the analysis and design of data flows equal weight to that of procedural or activity analysis and design.
 - Consequently, SADT uses its graphic language to construct two sets of diagrams: Actigrams (activity diagrams) and Datagrams (Data Flow Diagrams).
 - These two types of diagrams are not mirror images of each other, but rather complement each other. The datagrams are a form of data dictionary.
- Also unique to SADT is the concept of disciplined teamwork, in which people have specific responsibilities, and the method of communications between the team members is formally specified. Formally defined SADT team functions include:
 - Chief author or monitor.
 - Authors.
 - Commentators.
 - Readers.
 - Experts.
 - Librarian.
 - Technical Review Committee (TRC).
 - Project manager.
 - Instructor.

- "Authors" (analysts and designers) prepare "kits" consisting of actigrams and datagrams, for formal review by a number of commentators.
 - Commentators must respond in writing, and authors must either incorporate the comments in a new design or explain - in writing - why they reject the comments.
 - In large projects, a SADT librarian is in charge of the clerical details of keeping track of revision levels in the SADT diagrams, and of the distribution of "kits" and circulation of comments.
 - In cases of disputes that cannot be resolved between author and commentators, the Chief Author attempts to mediate.
 - If the Chief Author's intervention is insufficient, a Technical Review Committee hears the arguments and arbitrates.
 - However, the SADT team technique assumes that most communications will be written, and that meetings (such as TRC meetings) will be the exception rather than the rule.
 - Experience has shown that less than 5% of author/commentator disagreements ever reach the TRC level, according to SofTech.
 - Readers are not expected to comment, but only to read SADT diagrams for information.
- Thus the SADT team concept is reminiscent of the IBM Chief Programmer Team, except that:
 - The activity is analysis and design, not coding.
 - Meetings such as "structured walk-throughs" are discouraged rather than encouraged. Written comments are preferred to oral comments.

- Recently, it appears that the functions of "commentators" and "readers" in SADT teams have been merged.
- The SADT graphic "language" is somewhat more developed, more formalized and has more well-defined meanings than comparable graphics (e.g. Data Flow Diagrams of the Yourdon School).
- The same set of graphics is used in both actigrams and datagrams, but the meaning is different.
 - In actigrams, each box represents a specific activity: input data is represented by arrows arriving from the left; output data arrows leave from the right.
 - In addition, arrows arriving at the top of the box describe controls, which are data that constrain the operation or activity in some way. Control data are not transformed into output data.
 - Finally, arrows arriving at the bottom of the box are called "mechanisms," which describe the processor or device that performs the activity or stores the data.
 - In datagrams, the boxes represent data, arrows arriving from the left represent the activities that generate the data, and arrows leaving from the right represent the activities that use the data.
 - Datagram boxes can also have control activities (arrows arriving from the top) and storage mechanisms (arrows arriving from the bottom).
- There are additional constructs in the graphic language, including:
 - Bidirectional arrows to indicate stimulus-response (feedback) situations.
 - A scheme to assure consistency between parent and child diagrams.

- An activation/sequencing language to transform the functional representation into a sequence of events.
 - The advantages and disadvantages of SADT are summarized in Exhibit V-13.
6. HIPO (HIERARCHY PLUS INPUT-PROCESS-OUTPUT)
- HIPO, originally developed by IBM as a documentation tool, was released as part of the IPT (Improved Programming Technologies) package, which also included the Chief Programmer Team and Structured Walk-Through concepts.
 - HIPO consists of two types of charts:
 - "H" or Hierarchy charts, showing the top-down hierarchy of processes making up a system.
 - "IPO" charts, one for each box in the "H" chart, which show the inputs and outputs of each process, and describe the process itself.
 - These two chart sets are meant to be developed concurrently. As each process is outlined in the IPO chart, its component steps can be entered in the appropriate level of the "H" chart.
 - The graphics of the "H" and "IPO" charts are extremely simple, consisting of boxes and arrows (IPO) or boxes and connecting lines (H).
 - Natural language comments augment the graphical description on the IPO charts.
 - Because the HIPO symbology is rather general and does not become bound to a particular meaning until the natural language text is entered, HIPO charts are not limited to the description of coded procedures.

EXHIBIT V-13

TOOL PROFILE: SADT

NAME

SADT - Structured Analysis Design Tool

TYPE AND APPLICABILITY

Structured analysis; applies to the specifications (general design) phase

MAIN BENEFITS

- Force high level decisions early
- Makes visible competent staff
- Makes visible design progress
- Promotes communications, especially with Europe
- Improves reliability of code

ESTIMATED LCC IMPROVEMENT

Requirements and specifications	}	20%
Detail design		
Coding		
Test		
Maintenance:		35%-50%
Total LCC improvement:		30%-40% (User Report)

COSTS

\$12,000 for one week training for twelve people. Does not include computerized support, which is available on a T/S basis on Cybernet.

TRAINING

One user reports \$20,000 - 40,000 total cost to train ten "authors." Another said three months minimum.

LEARNING CURVE

Two to three large projects; 20-30% slow-down on first project.

DISADVANTAGES

- Requires much training
- Not enough automation
- No formal guidelines for function decomposition
- Covers only a small portion of project life cycle
- Long "front end" scares users and management

INTERACTION WITH OTHER TOOLS

Has been used in conjunction with Warnier-Orr, SDM/70, PRIDE; PSL/PSA; Nassi-Schneiderman; PDL

SUPPLIER

SofTech
460 Totten Pond Rd.
Waltham, MA 02154
(617) 890-6900

- Requirements analysis and structured design can use HIPO charts as the main graphic vehicle.
- It is possible to substitute a Nassi-Schneiderman chart for the "process" part of the IPO chart.
- HIPODRAW, a program to automate the maintenance of HIPO chart sets, is available from IBM.
- For large systems, manual updating of the chart sets becomes a significant effort.
- When HIPO charting is combined with structured design concepts, such as those of Constantine and Myers, a balanced structured methodology results.
- It is similar in conceptual depth to the ones from the Yourdon school, for example, or to the Jackson Program Design Methodology.
- HIPO's main advantage is its simplicity.
- HIPO's main disadvantages are:
 - Cumbersome maintenance unless supported by automatic aids such as HIPODRAW.
 - Without a corresponding conceptual framework, HIPO is merely a documentation aid.
 - There are no graphics or rules for the design of data structures or flows; the technique is primarily procedure-oriented. This limitation can be very serious, considering the power of a data flow analysis in arriving at an appropriate design.

7. STRUCTURED TABLEAU

- Structured Tableau is the name given by Donald R. Franz of Southwestern Bell (St. Louis) to a structured design methodology relying principally on an adaptation of the classical decision table construct.
 - The adaptation consists of adding to the classical decision table artifices that represent the structured programming constructs of sequence and iteration.
 - The classical decision table already implements the selection construct.
- While the technique is not supported by massive documentation or formally published texts, Franz has documented a number of cases that seem to show rather startling productivity improvement due to the use of Structured Tableau.
 - These were discussed earlier, in section B of this chapter.
- A summary of Structured Tableau's characteristics, advantages and disadvantages, is shown in Exhibit V-14.

8. SYSTEM DEVELOPMENT METHODOLOGIES (SDMs)

- System Development Methodologies (SDMs) are commercially available products, or sometimes internally developed ones, that are, in effect, extensive "cookbooks," or checklists, intended to guide and control the system development process, especially of large systems.
- An SDM is typically packaged in a set of binders or volumes that contain extensive text, graphics and forms.
 - These materials may also be augmented with some computerized aids, in which case the SDM is said to be partially automated.

EXHIBIT V-14

TOOL PROFILE: STRUCTURED TABLEAU

NAME

Structured Tableau

TYPE AND APPLICABILITY

Structured analysis; applies to requirements and specifications phases.

MAIN BENEFITS

- Completeness of design can be checked quickly
- Complexity of resulting programs can be predicted
- Eases task of getting user consensus on specifications
- Improved quality and maintainability of resulting programs

ESTIMATED LCC IMPROVEMENT

Requirements and specifications

Detail design

Coding

Test

Maintenance

Total LCC improvement: 60% to 76%

}

79-88% (See V-B-1)

50-67% (See V-B-1)

COSTS

No formal cost

TRAINING

5 days

LEARNING CURVE

15% of students become proficient right after training; 3% will never accept method. The rest improve with usage.

DISADVANTAGES

- Procedure-oriented - no facilities for data design
- No computerized assist

INTERACTION WITH OTHER TOOLS

None

SUPPLIER

Donald R. Franz
Southwestern Bell
915 Olive Street
St. Louis, MO 63101
(314) 247-5122

- Generally, a well-developed SDM will have some or all of the following elements:
 - Guidelines for system analysis and design, generally based on the utilization of structured concepts or a specific structured analysis/design methodology.
 - A well-defined, detailed, multiphased system life cycle description, with well-defined deliverables and/or review processes at the conclusion of each phase or subphase, and provisions for iterating in order to implement changes dictated by the review process.
 - Well-defined documentation standards, including forms.
 - Guides for planning and estimating resources, personnel and computing facilities.
 - Guidelines for acceptance testing.
 - Role definitions for work groups that are supposed to execute the SDM steps.
 - Project control mechanisms.
 - "Tutorials" - written or computerized explanations or demonstrations of the actual use of the SDM.
- An effective, well-designed SDM should have the following characteristics:
 - It must be easy to understand and use.
 - It must be flexible, or modular, so that small projects do not require the same level of detail appropriate for major projects.

- It must include provisions for the maintenance portion of the system life cycle, such as a change control mechanism to assure that documentation represents the current state of the product system.
 - It must take into account the fact that most new systems developed today are on-line systems rather than batch ones.
 - It must emphasize and encourage end user participation in the life cycle process.
 - It must match the characteristics and needs of the organization that uses the SDM.
- Needless to say, no single SDM can satisfy all of these requirements; but the list is still useful as a measuring stick or benchmark against which any proposed SDM could be measured.
 - Commercially available SDMs include, among others:
 - PRIDE/Logik from Milt Bryce & Associates, Inc.
 - SDM/70 from Atlantic Software, Inc.
 - SPECTRUM from Spectrum International, Inc.
 - CARA from Cara Corporation.
 - SYSTEMACS from Management & Computer Services, Inc.
 - All of the above are primarily or entirely manual procedures, but PRIDE/Logik from MBA is automated to a large degree (with the optional Logik and ADF, and the future GENASYS products). The next section gives a brief profile of this system.

- SDMs contribute to productivity through a number of mechanisms:
 - By providing a detailed "checklist" of tasks and procedures throughout the development cycle, the danger that some aspect of the system implementation might be overlooked or omitted from the estimation and planning functions is minimized.
 - By enforcing standardized documentation, clearer communications between users, developers and maintenance personnel results.
 - By providing a formalized, detailed life cycle, with defined deliverables and sign-off procedures at logical termination points, SDM makes possible project control and progress reporting.
 - By defining an orderly change control procedure, the chances that maintenance activities will not be reflected in the system's documentation are minimized.
 - The various tools employed by the particular SDM, such as structured design, automated data dictionary, etc., also contribute to productivity in their own right.
- The main difficulty with SDMs is this:
 - If an SDM is very complete, then its applicability to a given organizational environment may be limited. In fact, some organizations buy an SDM only to find that they must replace large chunks of it with home-brewed procedures to fit local needs.
 - Detailed SDMs are also hard to master: the volume of textual and tutorial materials (25 volumes in one case!) overwhelms the users.
 - On the other hand, if the SDM is designed to be flexible, so that it may be customized to a particular operating environment, it will generally

be incomplete without the addition of locally created procedures and forms.

9. PRIDE/Logik-ADF

- This SDM, marketed by Milt Bryce & Associates (MBA) of Cincinnati, consists of three separate products:
 - PRIDE (Profitable Information by Design) is a manual SDM. It is the base for the other optional products.
 - Logik (Logical Organizing and Gathering of Information Knowledge) is an automated data dictionary system, replacing a manual DD in PRIDE.
 - The combination of PRIDE/Logik is also called ASDM - Automated System Design Methodology.
 - ADF (Automated Design Facility, not related to IBM's ADF) is also an optional product that further automates the operation of ASDM.
 - Further automation will take place in the future, when the code-generating system GENASYS will be integrated to work with ASDM/ADF.
- PRIDE is the basic, manual portion of the methodology.
 - It is contained in a four-volume binder set.
 - It defines a life cycle consisting of nine main phases, with two sub-phases in phase 4.
 - It uses a structured design approach, based on component decomposition.

- Design is to be carried out top-down ("explosion"), while implementation is to be done bottom-up ("implosion").
- The procedures apply to non-EDP systems as well.
- Data are viewed as resources to be managed. The management mechanism is a manual data dictionary/directory in which specifications of all data items are kept. (This is the portion that the Logik option automates.)
- The DD also contains a cross-reference system for all systems, procedures and programs that use the data items listed in the DD.
- Manual project control, planning and estimating features are included.
- Both main projects and modification/enhancement projects can be handled.
- The system design tools, specification schematics, flow diagrams, and so on, used in developing the system, are considered to be the system's documentation.
 - In PRIDE, data documentation and process documentation are separate.
- An ECC (Engineering Change Control) system is used to assure that system changes are reflected in the documentation.
- Logik is a computerized data dictionary system that replaces the manual equivalent in PRIDE. It is an optional, extra-cost item.
 - Logik recognizes five levels of system components and six types of data components.

- Logik allows the specifications of relationships ("attachments") between data components and system (procedure, activity) components.
- The meaning and use of each data item is described by an installation-defined set of Logical Data Attributes.
 - . These are used as "keywords" to prevent duplication and to permit searching by content.
 - . Logik issues an error message whenever it detects two data items having the same set of logical attributes.
 - . Searches can be initiated for data items with specific attributes, or for items that come close to a given set of attributes.
- A variety of reports ("system component catalogs") can be automatically produced, based on the content of the data dictionary.
 - . These reports are regarded as a major portion of the system documentation.
- Logik is supported by its own set of three manuals.
- The ADF option is supposed to further automate some aspects of the PRIDE/Logik (ASDM) combination.
- MBA has released very little information on ADF, and the data that have been released are vague.
- MBA claims that ADF automates most of phases 1, 2, 3 and 4; it appears that ADF is an automated requirements language, somewhat like PSL/PSA (detailed later in this chapter).

- PRIDE users that INPUT interviewed did not have the ADF option and were reluctant to supply details due to confidentiality agreements with MBA.
- MBA has been involved in a long-standing law suit, charging that Arthur Young & Co. and the Harley-Davidson Division of AMF misappropriated PRIDE trade secrets in developing their own SDM; hence, apparently, the reluctance to furnish details on new products.
- Future plans call for the GENASYS code-generating system from Generation Sciences International (San Francisco) to be integrated with PRIDE/Logik-ADF to further automate the life cycle. This is planned for 1981.
 - GENASYS is expected to automate the remainder of phase 4 as well as phases 5 and 6.
 - When integrated with the MBA system, GENASYS will produce PL/I or COBOL code from system specifications ("computer run book") produced by iterative processes specified in PRIDE/Logik-ADF.
- A summary of PRIDE/Logik-ADF characteristics, advantages and disadvantages, is given in Exhibit V-15.

10. REQUIREMENTS LANGUAGES

- Requirements languages are systems of semantics and syntax specialized for describing the functional requirements of systems, especially (but not exclusively) software systems.
 - These are not simulation or modeling languages; they deal specifically with the description of requirements.
- The motivation for the development of such languages is that, by casting the system's specifications and requirements in machine-parsable form, it is

EXHIBIT V-15

TOOL PROFILE: PRIDE/LOGIK-ADF

NAME

PRIDE-Logik/ASDM with ADF

TYPE AND APPLICABILITY

SDM; applies to all life cycle phases

MAIN BENEFITS:

- Data dictionary and cross-reference
- Flexibility - can enter life cycle at any phase
- Good documentation
- Significant aid to maintenance

ESTIMATED LCC IMPROVEMENT

Requirements and specifications: 25% penalty

Detail design: 25% penalty

Coding: 20% (with GENASYS)

Test: 30%

Maintenance: 35%

Total LCC improvement: 25%
(INPUT estimate)

COSTS

\$25,000-55,000 one-time charge

TRAINING

Eight days provided by vendor

LEARNING CURVE

Approximately one month to gain mastery; lengthens development time on first two projects

DISADVANTAGES:

- Steep learning curve - doubles development time on first try
- Project control is not automated
- Long "front-end" scares users and management

INTERACTION WITH OTHER TOOLS:

GENASYS - a COBOL code generator (planned for 1981)

SUPPLIER

MBA Inc.
1248 Springfield Pike
Cincinnati, OH 45215
(513) 761-8400

possible to automate several important aspects of the functional specifications process; in particular, the ability to:

- Perform certain consistency and completeness checks.
 - Produce, on demand, documentation in various formats that reflect the latest state of the design process.
 - Produce progress reports automatically.
- Two examples of requirements languages, along with the computer programs designed to process the "programs" written in them, are:
 - PSL/PSA (Problem Statement Language/Problem Statement Analyzer) from the ISDOS (Information Systems Design Optimization System) project at the University of Michigan.
 - RSL/REVS (Requirements Statement Language/Requirements Engineering and Validation System), developed by TRW for use at the Ballistic Missile Defense Technical Analysis Center (BMDTAC).
 - In some respects, BIAIT (Business Information Analysis and Information Technology), developed by Donald C. Burnstine of BIAIT International, can also be regarded as a requirements language, although it is not automated.
 - PSL/PSA appears to be the better-known and more widely used of the three languages; the remainder of this section is dedicated to a brief description of PSL/PSA.
 - PSL is a language for describing "systems."
 - "Systems" in the PSL sense consist of "things" (objects), which may have "properties" associated with them.

- "Properties" may have "property values" assigned to them.
- "Objects" may be interconnected or interrelated in some ways; these connections are called "relationships."
- All of the above are very general and could apply to almost any system; to particularize this for an information system, a limited number of appropriately predefined objects, properties and relationships are used.
- The objective of PSL is to describe, in machine-analyzable form, as much as possible of the information typically found in functional requirements or overall design documents. More specifically, PSL recognizes certain types of objects and relationships that permit the following aspects to be described:
 - System I/O Flow: the interaction of the system with its environment.
 - System Structure: the hierarchy of objects comprising the system.
 - Data Structure: the relationships that exist among data items used or manipulated by the system, or as seen by the end users.
 - Data Derivation: the specifications that describe which data are manipulated or used by which processes, and how the output data are derived.
 - System Size and Volume: the factors that influence the volume of processing the system will do.
 - System Dynamics: how the system behaves over time.
 - Project Management: identification of people involved and their responsibilities, schedules, etc.
- PSA is the computer program responsible for:

- Parsing the PSL "programs" to extract the appropriate items required for all other PSA functions.
 - Storing the system's description in a data base.
 - Performing consistency checks.
 - Producing various reports.
- The reports obtainable from PSA include:
 - Data Base Modification Reports: records of changes made to the data base as analysts continually enter new specifications, requirements and design information.
 - Reference Reports: displays of the data base in various useful formats. For example:
 - Name List: displays all objects in the data base, along with their properties and the date of the last change.
 - Formatted Problem Statement Report: shows all properties and relationships associated with a specific object.
 - Summary Reports.
 - Data Base Summary Report: provides project management information by displaying the totals of various types of objects in the data base.
 - Structure Report: displays complete or partial hierarchies of objects.
 - Picture Report: shows the data flows in a graphical form.

- Analysis Reports: provide various types of analyses of the information in the data base. For example:
 - . Content Comparison: analyzes similarity of outputs and inputs.
 - . Data Process Interaction: shows gaps in the data flows and detects unused data objects.
 - . Process Chain Report: displays the sequence of events and processes in the system; i.e., the system's dynamic behavior in a graphical, flow-chart format.
- Some of the advantages of PSL/PSA are that it:
 - Improves the quality of the design by promoting precision in the requirements description, and by performing automatic consistency and deficiency checks, which are tedious and hard to do manually.
 - Permits the display of the most up-to-date state of the design by organizing the system description in a data base.
 - Promotes the detection and early removal of design errors.
- Probably the most significant current disadvantages of PSL/PSA are that:
 - PSA maintains a private data base/data dictionary, which is separate from the target system's data base/data dictionary.
 - The system is currently batch-oriented; i.e., PSL inputs and PSA reports must be handled as batch operations.
- A summary of the characteristics and benefits of PSL/PSA is shown in Exhibit V-16.

EXHIBIT V-16

TOOL PROFILE: PSL/PSA

NAME

PSL/PSA

TYPE AND APPLICABILITY

Requirements language; applies to requirements and specification phases

MAIN BENEFITS

- Central repository of all information about a system, including data, their attributes and structure, inputs, outputs, etc.
- Good documentation, produced automatically
- Promotes good communications within project team and with users
- Eases maintenance

ESTIMATED LCC IMPROVEMENT

Requirements and specifications - 20% (penalty)

Detail design - 14%

Coding - no change

Test - 20%

Maintenance - 10%

Total LCC improvement: 10% (INPUT estimates)

COSTS

\$15,000/year

TRAINING

Five days

LEARNING CURVE

One week for librarians, 2-3 months for system designers

DISADVANTAGES

- Lack of on-line facilities
- Lack of interaction with target system's data dictionary
- No preprinted forms for each object type

INTERACTION WITH OTHER TOOLS

AT&T has interfaced PSL/PSA to a data dictionary

SUPPLIER

Dr. Daniel Teichroew
Department of Industrial Engineering
231 West Engineering Building
University of Michigan
Ann Arbor, MI 48109
(313) 763-2238

- Although the system is nominally free and can be obtained for the asking from the ISDOS project at the University of Michigan, in practice such access is conditional on the applicant's becoming a "member of the ISDOS research effort."
- Members are expected to contribute \$15,000 per annum to the University.
- Members can, but are not required to, participate in the research effort and enhancement process of PSL/PSA.

II. PDL

- PDL (Program Design Language) is an example of a class of tools often called "pseudo-code."
- PDL, developed and marketed by Caine, Farber & Gordon, Inc., of Pasadena, CA, is a "pidgin" language in which natural English vocabulary is combined with the structured programming syntax of sequence, iteration and selection.
- In effect, a PDL "program" is actually a flow chart of a process that uses only structured programming constructs.
 - Unlike an actual program, the level of detail in a PDL "program" is up to the user.
 - PDL can be thought of as "Structured English."
- The important point about PDL, which it shares with other pseudo-code systems, is that system descriptions expressed in PDL can be quickly and straightforwardly converted to actual code in the selected target language.
- A computer program to process PDL descriptions ("segments") is available, and is also called PDL (or the PDL processor).

- Output from the PDL processor is indented to show nested levels of control.
 - A table of contents is automatically generated from the segment titles provided by the user.
 - "Trees" show the hierarchy of segments by automatically recognizing references from one segment to another. A separate tree is produced for each root segment.
 - Data and segment indices list alphabetically the names of all data items and segments identified as such by the user.
- Despite the limited functions performed by the PDL processor, the suppliers claim a range of benefits that is impressive, including significant reductions in computer utilization per line of code developed.
 - The main disadvantage is that there are no facilities for describing data structures and their interrelations with procedures. PDL is completely procedure-oriented.
 - Exhibit V-17 summarizes the characteristics, benefits and limitations of PDL.

12. REUSABLE CODE

- In the late 1960s, when interest in the emerging "software engineering" was building up, a number of suggestions were made indicating that, to improve programming productivity, the construction of programs must be made to more closely resemble the design of electronic hardware systems (or any hardware, for that matter).
- Designers of hardware systems rarely need to create a completely new system from scratch.

EXHIBIT V-17

TOOL PROFILE: PDL

NAME

PDL - Program Design Language

TYPE AND APPLICABILITY

Pseudo-code; applies to detail design and coding phases

MAIN BENEFITS

- Reduced computer utilization per line of finished code
- Decreased debugging effort
- Increased programmer productivity measured in lines of code
- Automated cross-reference
- Systematized archiving

ESTIMATED LCC IMPROVEMENT

Requirements and specifications

Detail design

Coding

Testing

Maintenance: No change (INPUT's estimate)

Total LCC improvement: 25%

} 75% improvement (manufacturer's documented experience)

COSTS

\$3,000

TRAINING

Less than one week

LEARNING CURVE

Short, assuming previous exposure to structured concepts; long otherwise

DISADVANTAGES

- Oriented to procedural design only - no data design facilities
- Batch environment only
- Relatively limited coverage of life cycle

INTERACTION WITH OTHER TOOLS

DES, an integrated software design system (Graham et al., Communications of the ACM, February 1973) is based on PDL

SUPPLIER

Caine, Farber & Gordon Inc.
750 East Green St.
Pasadena, CA 91101
(213) 449-3070

- Instead, they build complex systems from commercially available, prepackaged subcomponents, such as today's integrated circuits, which perform some common functions in a standard way.
- In the same manner, designers of software systems should have available libraries of prepackaged basic function programs, from which complex systems could be built.
- Unfortunately, a "standard software module" industry has not developed so far.
 - Many objective reasons for this can be cited, including the proliferation of incompatible hardware, at least in pre-System/360 days, and of incompatible operating systems, even today.
 - However, it is probable that the real reason is the belief, held especially by programmers but also by some EDP management, that each application, each program and each data field is unique; that is, the idiosyncracies of each company's business practices preclude the use of any "general-purpose" code.
 - Perhaps a more appropriate name for this belief is "NIH Complex" ("Not Invented Here").
- It is interesting to note that in scientific programming, much progress has been made in this direction.
 - Libraries of common mathematical functions, statistical routines, linear programming packages, etc., are standard tools in any installation supporting scientific computing.
 - However, much less has been achieved in standardized "drivers" for these routines.

- The increasing acceptance of packaged application systems as a substitute for in-house development is an indication that some standardization is possible.
- In recent years, there has been an increased interest in standardized software modules to perform common, business-oriented DP functions.
 - Kapur & Associates, Inc., a Danville, CA, consulting firm, has been advocating the use of prepackaged, standardized modules for a number of years, and has developed packages called BUMP (Building Universal Model Programs) and GENIUSS (Generalized Installation Uniform Source Statements).
 - It reports 50% to 80% productivity improvements.
 - Raytheon Missile Systems Division has developed a system of pre-packaged, standard-function modules that they call simply "reusable code."
 - It reports 40% to 60% productivity improvements.
 - Raytheon Service Co. is in the process of turning this into a commercially offered product, probably under the name Universal Productivity Package: it was scheduled for release in the last half of 1980, but as of mid-October the system had not yet been released.
- Both Kapur and Raytheon begin by noting that practically all business applications can be classified into a very small number of activity types.
 - Kapur identifies them as data selection/edit, data extraction, data update and report.
 - Raytheon's classification is data select/edit, data sort, data update and report. It excludes sorting from the UPP, since this is provided as a packaged system by the computer manufacturer.

- In the Kapur system, the standardized modules consist of common logic structures for selected, common functions such as sequential file update.
 - These logic structures are expressed in terms of hierarchical diagrams, Warnier-Orr charts and Nassi-Schneiderman charts.
 - In addition, these structures are partially implemented as COBOL and PL/I programs, with several items left open for the programmer to fill in for the particular needs of the application.
- The Raytheon system has about 1,500 prepackaged modules, of which:
 - Some are completely coded into "functional modules" (COBOL programs) that are stored in a central library and can be retrieved by up to five levels of search keys;
 - Some are incompletely coded logic structures, with areas left undefined, to be filled in as required.
- The Raytheon system also includes a number of additional tools:
 - A COBOL reformatter that standardizes the COBOL source code generated by the programmer as fill-ins to predefined logic structures, or as higher-level code calling on prewritten functional modules.
 - A test data generator.
 - A program revision control system for maintaining multiple versions of programs in development.
 - A debugging tool, a file comparison tool and other tools.
- Both Kapur and Raytheon report remarkable productivity improvements, especially for entry-level programmers.

- They reach a high level of productivity in three months, according to Raytheon.
- Other, obvious advantages of the reusable code system include:
 - Reduction in overall (not initial) coding effort.
 - Reduction in maintenance effort because:
 - Precoded modules that have been used a number of times become fully tested.
 - The common programming style of precoded and semicoded modules eases the task of understanding "foreign" code.
- Exhibit V-18 summarizes some of the characteristics of the Raytheon reusable code system.

13. MENU-DRIVEN PROGRAMMING

- Menu-driven programming is INPUT's term for systems in which some or all of the traditional, procedure-oriented programming tasks are replaced with "fill-in-the-blanks," interactive sessions with the user.
- Conceptually, menu-driven programming is halfway between the reusable code approach and packaged application systems.
 - It generally consists of a group of skeleton, prepackaged tasks, usually backed by a data base management system and an inquiry/retrieval system.
 - Through the interactive sessions, the user supplies additional specifications and code to "flesh out" the skeletons.

EXHIBIT V-18

TOOL PROFILE: RAYTHEON'S UPP

NAME

UPP - Universal Productivity Package

TYPE AND APPLICABILITY

Reusable code system; applies primarily to detail design, coding and test phases

MAIN BENEFITS

- Reduced program development time
- Reduced maintenance
- Reduced requirement for highly-skilled people
- Usage statistics are automatically provided

ESTIMATED LCC IMPROVEMENT

Requirements and specifications	- no change	
Detail design: 40-60%		} (Vendor's estimate)
Coding: 40-60%		
Test: 40-60%		
Maintenance: 40-60%		
Total LCC Improvement:	38-57%	

COSTS

\$50,000-100,000 (estimated)

TRAINING

Five days for experienced analysts, plus three days for supervisors. Two-month comprehensive program for entry-level programmers.

LEARNING CURVE

Quite short; 1-2 months or 2-3 uses of system

DISADVANTAGE

- Current product is limited to batch environment (interactive support planned)

INTERACTION WITH OTHER TOOLS

ADR's Roscoe, Librarian now; TSO and Panvalet planned

SUPPLIER

John Cooper
Raytheon Service Co.
(617) 273-4655 Ext. 30

- The term "non-procedural programming" has been applied to menu-driven systems in some literature. "Menu-driven" avoids confusion with such non-procedural languages as SIMSCRIPT.
- Some examples of menu-driven programming systems follow:
 - ADF (Applications Development Facility) is an IBM IUP (Installed User Program) that creates a menu-driven environment in conjunction with IMS.
 - DMS, variously known as Development Management System or Display Management System, is available in several, not entirely compatible, versions for the 8100 under the DPPX operating system, the 3770 and 3790, and for DPD operating systems, including VM/CMS, OS versions and DOS/VS. It creates a menu-driven programming environment in conjunction with CICS/VS. (The 8100 version can also work with IMS/VS at the host).
 - Nocode, a menu-driven programming system, operates on General Automation minicomputers.
 - TAPS Transaction Processor (TTP) from Decision Strategy, Inc., now part of Informatics, is a menu-driven programming system available in a variety of versions to run on different hardware systems, including IBM (OS and DOS) mainframes, Interdata, DEC, HP, Prime and Harris minicomputers.
 - In the Wang 2200 and VS computers, all job control statements have been replaced by fill-in-the-blank sessions with the user at an interactive terminal; application coding, however, is still done by traditional, procedural coding.

- The advantage of such system is the speed and ease with which new applications can be developed and changed, frequently cutting weeks-long effort to a matter of hours or days.
- The disadvantage is that the prepackaged skeletons may fit only a certain class of problems. Employing user exits to add "own code" routines invariably complicates the design to the point where use of the menu-driven portion is often not worth the effort.
- In typical use, the menu-driven system might be employed for the class of small applications to which it is best suited, while major systems that do not fit the skeletons well are developed by traditional procedural coding.
 - Experience will show where the dividing line should be placed.

D. FUTURE DIRECTIONS

- The design, implementation and maintenance of information processing systems involves a number of creative activities that are highly dependent on human intelligence, experience, skill and resourcefulness.
 - These activities are not amenable to automation, so improving their productivity must rely on other techniques: personnel selection, training and motivation, organizational and environmental factors, and so on.
- Many aspects of design, implementation, validation and maintenance are basically clerical, bookkeeping or repetitive tasks.
 - Such tasks could all be automated, with productivity improvements expected as a result.

- In fact, most such tasks have already been automated to some degree, as the previous brief review of tools and aids has clearly demonstrated.
- The underlying, common problem of all existing tools and aids is their ad hoc nature; they each address a specific area independently of other related areas. For example:
 - Existing requirements languages have no interface to the detail design and implementation phase.
 - Structured design and analysis methodologies have little or no transition mechanism into coding.
 - Structured design and analysis methodologies address the entire life cycle, but only as a "checklist;" i.e., without tools to address the substance of the life cycle activities.
- Another major problem is the scarcity and relative ineffectiveness of techniques, tools and aids to influence the maintenance activity directly.
 - Useful tools not now commercially available would be:
 - A global cross-reference generator, operating across the entire application system of main program, called subroutines, job control statements, utility parameter lists, etc.
 - A mechanism to retroactively populate the data dictionary (related to the cross-reference generator).
 - A global documentation generator/editor that would facilitate structured, "reusable" documentation.
 - A regression test harness or driver.

- The absence of these tools leaves a rather big gap in productivity improvement, since INPUT estimates that the majority of the life cycle costs, at least of major projects, are due to maintenance rather than development activities.
- What is clearly needed is a system to integrate the automation of all life cycle activities into a uniform, compatible and communicating set of techniques and tools.
- Using the existing state of the art as a base, without requiring any revolutionary breakthroughs, it is possible to visualize such a system, and to list its major features and characteristics.
 - It will have an interactive "workbench" to serve the needs not only of coders, but also of system specifiers, system designers and maintenance personnel.
 - The system specifiers and designers, working interactively via their unique "workbench," will accumulate the specifications and overall design of the system, cast in a "requirements language."
 - The requirements language processor will store the elements of the system specifications and design in the same data base and data dictionary that will eventually be used for the operation of the system being designed.
 - The requirements language processor will perform consistency checks and produce documentation to serve the rest of the system's life cycle.
 - The requirements language processor will produce one or several alternate "trial" designs, and will forecast their performance under a range of operating assumptions.

- Specifiers and designers will think in terms of a uniform, company-wide structured design and analysis methodology.
- Detail designers and coders, supported by their own unique "work-bench," will convert the procedures and data designated in the specifications phase into executable code, using a very high-level language supported by the appropriate code generator.
- They will be able to search a large library of previously developed standard functions and skeletons by context, to determine which of the existing modules and skeletons already satisfy portions of the new design.
- It may even be possible to automate such searches directly from the system specifications output of the requirements language. This type of automated system might look for degrees of compatibility and recommend modules that match desired profiles to a given degree (e.g., "looks like it will do 75% of what you want").
- The designers and coders will be trained in structured design and coding and will therefore produce code in a uniform style; that is, they will do so only if the functions they want are not in the reusable code library.
- Functions that appear to be of general usefulness, but that are still missing from the library, will be marked for especially exhaustive testing, because they are candidates for certification as "reusable" in future systems.
- For relatively small, "run-of-the-mill" systems, designers and coders will turn to a number of menu-driven, prepackaged systems.

- Some end user requirements will not even reach the DP department, because the installation's data base management system, with its associated query/retrieval language, will allow them to create mini-applications without assistance.
 - Throughout the entire process, the activities of all personnel will be guided by a checklist provided by a system development methodology.
 - Progress reports and personnel/resource loading data will be automatically produced by the system's automated components. For this purpose, these components will be able to exchange data.
 - Documentation will be produced automatically from the data in the requirements data base and data dictionary.
 - Both planned and unplanned enhancements to the system will be handled as "projects," although they will not require the complete SDM checklist.
 - There will be an automated "version control" system that permits binary modules to be unambiguously identified with the source code and design versions that produced the binary modules. This system will also control what goes into the "production" version of the system.
- Note that all of the above capabilities already exist in some automated form or another.
 - What is missing is the "glue" that binds the individual techniques into a uniform, compatible, communicating system.
 - The other missing element is the automation of program validation and checking.
 - Clearly, future progress must then proceed along two lines of attack:

- Developing integrated systems of uniform, communicating tools.
- Developing effective, simple tools to automate program validation and checking.
- Some progress towards these goals is already taking place. Two examples are appropriate:
 - The gradually increasing automation of the PRIDE system development methodology.
 - The CPDS system, in use by Chevron.
- When PRIDE becomes fully integrated with Logik, ADF and GENASYS, it will have many of the elements of an integrated system:
 - A system development methodology to envelop the entire life cycle.
 - A requirements language with its computerized processor (ADF).
 - A computerized data dictionary system (Logik).
 - A high-level language with associated code generator (GENASYS).
- The main elements of an integrated system still missing from PRIDE/Logik/ADF/GENASYS are:
 - The data dictionary which is still "private" to the SDM and needs special processors to interface with the actual application system.
 - The project control and management system, which is still manual.
 - The fact that the system does not include provisions for reusable code libraries or menu-driven, prepackaged applications.

- Chevron has reported on its in-house Chevron Program Development System (CPDS), which also has some of the prerequisite capabilities of an integrated system.
- The specific environment addressed by the Chevron system is as follows:
 - Program development, testing and maintenance occurs in an interactive environment, controlled by a VP/CSS operating system (commercially marketed by National CSS, similar to IBM's VM/CMS).
 - Production runs take place in an MVS/IMS environment, on hardware that is physically separate and remote from the NCSS timesharing system.
- This leads to both problems and opportunities:
 - Testing of IMS-based applications in the timeshared environment requires special artifices.
 - However, the separation of development and maintenance from production is beneficial in several respects.
- CPDS does not address the requirements and specifications phases.
 - The system concentrates on the detail design, coding and testing.
 - For these activities, the system integrates tools to a remarkable degree.
- More specifically, CPDS has the following features and capabilities:
 - An interactive, "workbench" environment for program development, testing and maintenance.

- Use of a very high-level language, PL/X, with the associated code generator that translates this language into PL/I code.
- An automatically enforced development sequence that requires the following activities in the following order:
 - Definition of data types and data fields (via RAMIS data base system) and data values (via SCRIPT).
 - A special "structure generator" (TS/X) operates on these definitions to create the appropriate PL/I DECLARATIONS and stores them in %INCLUDE files for later incorporation into PL/I programs.
 - Documentation of the program about to be created must then be entered via the SCRIPT text-processing system. No code may be entered unless and until the documentation is entered.
 - The program is written in the high-level language (PL/X). From this, a special processor generates actual PL/I code along with the appropriate data base interfaces.
 - The PL/X processor also extracts from the program the data for a global cross-reference system, also expressed in SCRIPT.
 - The PL/I compiler then generates the code, incorporating the data definitions from the %INCLUDE files.
- Automatic project-management data are collected by the various components of the system. Reports include:
 - Personnel utilization.
 - Machine resources used.

- The status of each module in the system.
- Status is categorized as one of the following conditions:
 - Identified function.
 - Documented function.
 - Coded function.
 - Tested under the T/S environment.
 - Tested under the MVS/IMS environment.
 - Included in the production system.
- The PL/X processor enforces structured discipline both by providing the structured constructs as macros, and especially by deleting all labels so that GO TOs have no place to "land."
- Testing in the T/S environment requires an IMS simulator and a special version of the PL/X processor.
- As programs are tested, status is automatically recorded in the high-level source code, permitting quick determination of those elements of the program that have not undergone testing.
- Some of the major shortcomings of the CPDS system are:
 - No interface to a requirements language.
 - No support of reusable code or menu-driven programming.

- While neither CPDS nor PRIDE/Logik/ADF/GENASYS are currently fully integrated systems in the sense described earlier in this section, there are clear efforts to move in the right direction.

VI SELECTED CASE HISTORIES

VI SELECTED CASE HISTORIES

- Organizations described in this chapter are real, but names and identifying characteristics have been deleted or modified to protect proprietary interests.
- Both successful and unsuccessful histories have been included.
 - Cases 1-4 concentrate on the history of the organization.
 - Cases 5-10 focus on the history of, and experience with, a specific technique.

A. CASE HISTORY #1 - SUCCESSFUL STAGE TWO MANUFACTURER

I. THE COMPANY

- Case 1 is a Fortune 200 company with multinational sales exceeding \$2 billion. The company is engaged primarily in the manufacture and distribution of pharmaceuticals.
- Return on revenue and equity, and revenue growth have been among the best of Fortune 500 companies over the past 10 years.
- With over 30,000 employees, the company enjoys an excellent reputation:

- As a leader in its field.
- As a well-managed company.
- On Wall Street, as a sound-growth stock.

2. EDP ORGANIZATION

- The Company has reached stage one, "Control," by centralizing MIS as a function at corporate headquarters.
- The corporate data center has two large-scale IBM 370 computers that are interconnected in a star network to approximately 20 smaller EDP systems in distribution centers and plant sites around the U.S.
- There are some 30 separate, small EDP systems in major countries overseas.
- The corporate EDP facility employs 350 people, 200 of whom are professional analysts/programmers for commercial systems.
- Applications development and programming for overseas operations are executed overseas under the control of the U.S. corporate international group.
- The EDP organization is moving to stage two, "Quality." For some time the organization has been at the leading edge of technology with sophisticated applications and widespread use of software productivity aids and software packages.
- The EDP staff is mature and diversified in experience, and it is aligning on the basis of business functions. Professionally the company has a very good EDP reputation and is considered a good place to work.

- The company is developing a more extensive human resources system to provide its MIS personnel better promotional potential through formal career path planning.

3. APPLICATION TYPES

- In stage one, the company is standardizing its applications across operating divisions both in the U.S. and abroad.
- The company is standardizing on IBM Systems 34 and 38 for all smaller overseas facilities, and IBM 4300s in larger ones.
- Development of standardized business applications (sales, inventory, billing, etc.) for the southern hemisphere has been centralized in Lima, Peru. A similar effort is under way in Paris for Common Market countries.
- A standardized cost accounting system has been implemented worldwide.
- An overseas telecommunications network has been established to interconnect EDP systems in Common Market countries with the corporate EDP center.

4. EDP MANAGEMENT DESCRIPTION

- The company, a heavy user of automated systems, spending in excess of 1% of its revenues on EDP, has recognized that MIS is vital to its business operations.
- In an effort to stem corporate-wide payroll increases averaging 25% per year, the company installed a new corporate MIS director in 1979.
- Operating between stages one and two, the MIS director's charter has been to:
 - Curtail MIS budget increases.
 - Obtain more deliverables for the same development dollar.

- Improve overall communication and participation between the users and MIS.

5. CONTROL MECHANISM

- In an effort to improve the control and quality of its software, the company chose, in late 1979, Improved Systems Technology, Inc., to teach and consult, during the first year of use, the STRADIS System Development Methodology.
- To date, 40 analysts and programmers have attended two five-day courses, with additional user personnel attending an abbreviated course.
- One large (five person year) project (accounts payable) and several smaller projects are well under way using the STRADIS SDM.
- To accompany this effort, the company early in 1980 installed PC/70 to plan and report on MIS project development.
- Management, now more interested in productivity improvement, is more active in MIS affairs.
- In connection with its stage two, "Quality," control mechanisms, and moving toward stage three, a corporate steering committee has been formed. Composed of senior operating executives, the committee meets to approve corporate policies related to distributed processing, MIS expense chargeback and overall MIS goals and objectives.
- A recently established MIS Steering Committee, composed of operating division executives, approves development projects and establishes development priorities.

6. PERFORMANCE CHARACTERISTICS

- No longer in chaos, but still struggling to slow cost and schedule overruns and control backlog, the company can be considered to have entered stage one, "Control."

- The company expects that the combined use of automated project controls and a system design methodology, along with the recently established steering committees, will improve both planning and performance.

7. QUALITY CONTROL

- Through use of the system design methodology, the company is just beginning to focus attention on quality control.
- Moving toward stage two, "Quality," the new company-wide accounts payable project, now under development using STRADIS, considers quality assurance during the design process. Formal quality assurance design review, and test plan criteria and procedures, are being separately developed concurrent with the application implementation.

8. USER INVOLVEMENT

- Moving toward stage two, the company is fostering greater user involvement in the software development life cycle.
- The new position of Information Resource Manager was created late in 1979 to improve liaison between users and MIS.
- Five senior analysts located in MIS, one for each major segment of the company, are interacting with users in systems planning, requirements planning, MIS performance and MIS/user communications.
- In the international area, plans are underway to encourage users to utilize new IBM systems application development software to develop simple, local systems.
- The company is advocating the use of personal computers (such as the Apple II) in user departments for individualized applications.

- Users have been encouraged to utilize the recently installed Inquire capability of the S 2000 DBMS systems for data access and for one-time reports.

9. PROGRAMMING RESOURCES

- In stage two, the company offers programmers access to on-line terminals for program development.
- Adequate priority is given to assure access to computer resources for program development.
- Plans are underway to improve efficiency by providing each analyst/programmer with his or her own terminal.
- The company has made a conscious decision to make greater use of outside consultants and contractors to assist in program development.
- Para-professionals are being trained and used to assist in more routine programming support functions.

10. PRODUCTIVITY TOOLS

- The company uses a wide variety of productivity tools indicative of stage two development. Representative tools are TSO, S 2000 with Inquire, and structured design and programming techniques.
- Looking toward stage three, the company is considering ways to integrate the program development process. The company expects to decide on off-loading program development from the mainframes either through programmer workbenches (UNIX) or through the use of an IBM 4300.
- The company has made a conscious decision to buy more applications software.

II. INVESTMENT PAYOFF

- Having gained a measure of control over EDP planning and development, the company in stage two is experiencing selected benefits from its continued investment.
- Communication and overall user liaison have improved through establishment of Informance Resource Managers.
- More thorough system specifications and better user understanding have been realized by initial use of a structured design methodology. Documented deliverables are standardized. Currently increasing its system analysis investment, the company expects payoff later in the life cycle.
- Standardizing EDP hardware and consolidating applications development for Europe and South America has resulted in increased productivity in these locations.
- Through the implementation of automated project management (another investment), the company expects to achieve future payoff both in overall productivity and in user satisfaction with quality applications delivered on time.
- The company reports a significant improvement in EDP department morale.
- The company reports satisfaction with its progress to date in productivity improvement. Consensus is that many of the processes resulting in productivity improvement could well have been started sooner. Examples cited were the system design methodology and user involvement in the design process.

B. CASE HISTORY #2 - SUCCESSFUL STAGE ONE MANUFACTURER

I. THE COMPANY

- The company, among the Fortune 200, is a diversified manufacturer and distributor of chemicals, paper, food and other consumer products.
- Growth from within, as well as through mergers and acquisitions, has more than doubled sales of nearly \$2 billion in less than a decade, with corresponding increases in earnings of over 600%.
- With nearly 29,000 employees worldwide, the company is considered conservative in its approach to automation.
- With a low return on sales and investment, the company is not currently regarded as a capital growth opportunity by the investment community.

2. EDP ORGANIZATION

- With the centralization of MIS as a corporate function, the company has established "Control" in stage one.
- Over 300 personnel, 100 of whom are analyst/programmers, support all business divisions. The MIS staff is located at a large data center, which has a very large IBM mainframe interconnected to a smaller IBM system at corporate headquarters.
- Plans have been approved for a new corporate data center, consolidating the existing data center and corporate data center. The new center will house all the EDP development staff plus all IBM 158 and 3033 mainframes.
- Various small computers, some interconnected in a star network, are distributed among 20 plant and distribution centers throughout the U.S. Plans call

for connecting all data centers with the corporate center through a new data communications network.

- International DP facilities exist for some countries and markets.
- The company is considered to have an experienced DP group that is operating in a deliberate and conservative fashion.

3. APPLICATION TYPES

- Entering stage one, "Control," the company is beginning to standardize basic applications packages across operating divisions.
- The company has standardized on IBM System 34 computers for its international operations.
- Basic accounting, order entry and financial reporting applications packages for the Southern Hemisphere are being developed in Mexico City. A decision has been made to develop standardized applications for Common Market operations by a corporate group located in Europe.
- A project has just gotten under way to examine corporate data in a disciplined fashion to improve its consistency and to coordinate its uses in the company's systems and operations.

4. EDP MANAGEMENT DESCRIPTION

- Recognizing the important impact of information systems on operating efficiency, the company in 1979 established a formal, long-range planning activity in data processing. The initial 1979 long-range plan is being revised for implementation in the 1980s.
- Long-range business systems planning has also been introduced in the operating divisions, linked with the DP long-range plan.

- Because of the company's low return on sales and investment, there is a renewed drive to improve operating margins throughout the company.
- As a first step in progressing from stage one, "Control," to stage two, "Quality," the company has established objectives for improving EDP productivity:
 - Improve effectiveness of business systems planning.
 - Accelerate and improve information systems development, including end user involvement.
 - Improve efficiency and reliability of information processing services.
 - Make computerized information more easily accessible to end users.
 - Introduce the concept of data as a resource.
 - Integrate office automation with information systems development.
 - Expand automated control of manufacturing and environmental control processes.
- The company recently installed a career development system for MIS personnel.
- Training is provided to improve skills required for future projects. Technical skills are developed through Deltak courses.

5. CONTROL MECHANISM

- The company is beginning to move toward stage two by implementing a project life cycle concept. The structured design techniques of analysis, design, coding and implementation are being integrated with a systems design

methodology by modifying the "CARA" SDM originally developed by the Kraft Company.

- The company has selected Improved Systems Technology Inc. to train its professional staff in structured design techniques.
- In the last half of 1979, over 100 professionals attended several five-day courses in structured design methodology.

6. PERFORMANCE CHARACTERISTICS

- Just entering stage one, "Control," the company expects that by consolidating corporate U.S. activity in one data center, along with international systems in South American and European development groups, it will be better able to control schedule and cost overruns.
- Implementation of a formal systems design methodology and structured program design and development techniques are eventually expected to control backlog.
- Greater user participation in the planning process, and the use of more direct user data access, are the routes the company is taking to eventually reduce backlog.
- In the meantime, the additional control mechanisms have lengthened the development process.

7. QUALITY CONTROL

- Not yet at stage one, the company is at the point of considering quality assurance as it develops its in-house version of a system development methodology based on CARA.

8. USER INVOLVEMENT

- The company is properly beginning to encourage greater user involvement in the system design process, in an attempt to minimize the negative effects that stage one has on the user.
 - Courses are being developed for both user and MIS personnel to become more aware of each other's responsibilities and to foster joint working relationships. Structured design techniques and data base management are an integral part of the course material.
- Long-range planning functions have been assigned to user divisions to coordinate systems requirements and definitions with MIS, by means of the system development methodology (CARA).
- The strategy appears to be working, as user management reports greater satisfaction with systems definition and development through improved involvement with MIS.

9. PROGRAMMING RESOURCES

- The company is squarely in stage one by currently offering programmers some access to terminals for interactive program development.
- Plans for the new corporate data center call for increased availability of analyst/programmer terminals, approaching a 2:1 programmer/terminal ratio.
- Assured priority access for program development has been included in plans to upgrade mainframes at the new corporate data center.
- The company is looking toward the use of professional services and the possible use of industry-oriented software packages to assist it in automating its manufacturing and environmental control processes.

10. PRODUCTIVITY TOOLS

- Already using TSO, the company is just beginning to implement additional systems and software productivity tools, consistent with its stage one status, moving toward stage two slightly more rapidly than expected.
- Seventeen projects using structured design and development techniques are in various stages of analysis through implementation.
- The company is investigating the use of both query languages and improved report generator software, in order to be able to respond more rapidly and economically to users' requests for file information.
- The company is increasing its emphasis on procuring applications packages, where possible, as opposed to allocating resources for developing standard applications in-house.

11. INVESTMENT PAYOFF

- The company's general progress toward increased productivity has improved over the last two years.
- Improvements have centered around structured techniques, more effective planning and comprehensive training/career development.
- Use of structured techniques has already improved communication between users and MIS and has resulted in better specifications. The company, now squarely in stage one, "Control," expects an initial 10% gain in productivity, eventually increasing to as much as 20% in the project life cycle.
- Further investment in formal long-range planning, for both MIS and user areas, is expected to improve systems development and integration. Use of the SDM is already helping to make MIS personnel more effective on high-payoff projects.

- A formal skills evaluation program has been successful in reducing turnover and improving the professional staff's proficiency.
- Savings have been experienced in international operations through common applications development, producing a corresponding increase in users' satisfaction.
- The company expects to achieve significant future payoffs through:
 - Automated control of manufacturing processes.
 - Integrating office automation under MIS systems.

C. CASE HISTORY #3 - SUCCESSFUL STAGE THREE MANUFACTURER

I. THE COMPANY

- With multinational sales exceeding \$20 billion, this Fortune 50 company is primarily a process manufacturer and distributor.
- Return on equity has exceeded 14% AAGR over the last 10 years.
- The company has experienced explosive revenue growth in the last five years.
- With nearly 40,000 employees, the company is considered:
 - A sound-growth stock.
 - A well-managed multinational corporation.
 - A highly desirable place to work.

2. EDP ORGANIZATION

- Experiencing explosive growth in wholesale marketing operations, the company over the past five years has centralized responsibility for its EDP operations at its West Coast corporate headquarters.
- Three major data centers have a combined data processing capability equivalent to eighteen IBM 370/168s.
- Corporate Computer Services employs some 450 analyst/programmers, organized in stage two by business function, using program management concepts for project implementation.
- Development projects are organized in small chief-programmer groups comprised of a chief programmer, a project analyst and four to six programmers.
- Applications are broken down into projects of approximately five person years of professional effort.

3. APPLICATION TYPE

- Currently operating in stage two, the company has developed a number of data base-oriented applications that permit uniform, company-wide operations.
- Operational information generated through production and marketing applications is available to financial and management information applications, demonstrating cross-functional integration.
- Moving toward stage three, the company is planning to use its operational information data base for the strategic financial management and planning of its international operations.

4. EDP MANAGEMENT DESCRIPTION

- Recognizing the vital importance of electronic information to corporate vitality, the company has elevated EDP in the corporate organization.
- Directly reporting to the Executive Vice President of the Operations Group, the Director of Computer Services is involved in corporate strategic planning, in accordance with stage three management concerns.
- The Director of Computer Services has been given the charter to establish a company-wide methodology for definition and specification of advanced management information systems to implement company marketing and production activities.

5. CONTROL MECHANISM

- Operating in stage three, the company has established EDP Steering Committees at two levels. The Corporate Committee, which has the EDP director and corporate officers as members, sets overall goals and objectives, and sets priorities on a global basis.
 - Another steering committee at the operating level includes divisional managers or direct users. This committee accepts projects for implementation, setting and reviewing priorities during the year.
- The company has integrated a project control system with its program development system.
- The in-house program development system breaks a project into programs, and a program into modules (100 executable statements). The automated system tracks module development from design to completion, accounting for new module definition whenever it occurs.

- The company's experience has been that, shortly after the number of new modules tapers off, predictions concerning budget and schedule become reliable for project assessment.
- The program development system (PDS) is highly automated and covers the program design, test, implementation and maintenance stages in the system development life cycle.
- By requiring that documentation be an integral part of program development activities, rather than an afterthought, the PDS encourages comprehensive systems definition at the front end of a project.
- The company is actively investigating the use of a formal system design methodology that involves users with analyst/programmers, and that includes prototyping through menu-driven languages.

6. PERFORMANCE CHARACTERISTICS

- Although its backlog is very large (in excess of \$10 million in professional effort), the company believes that the in-house PDS has stabilized the situation.
- Realizing its inability to recruit sufficient numbers of professional staff to support growth, the company was forced to concentrate on improving the productivity of its analyst/programmers.
- Implementation of the in-house PDS has begun to move the company toward stage three. Professional productivity (from the program design stage on) has increased five to seven times over batch development. This increase has required substituting 2-2.5 times the former amount of processing resources, a trade-off the company has been quite willing to make.

7. QUALITY CONTROL

- Operating well into stage three, and quite advanced in its quality control procedures, the company has integrated program testing and quality assurance into its automated PDS.
- The PDS supports multiple levels of testing:
 - Initial top-down tests on the timesharing development system.
 - IMS batch terminal simulator testing.
 - Preproduction integration testing with IMS.
- The PDS accomplishes line-by-line certification of programs, ensuring that the logic of modules of code has been properly executed. The PDS automatically keeps track of the past four test runs, for regression test purposes.
- The PDS automates program release of the completed system (or module), from maintenance to production.
- The PDS maintains program documentation throughout the development and maintenance cycle.
- The company is implementing scheduled releases of program enhancements at approximately six-month intervals.
- The company has established a separate quality assurance group to ensure that reliability is considered a part of program design.

8. USER INVOLVEMENT

- User involvement is the area currently lagging farthest behind the company's general stage three posture. During the past year, the company has encouraged greater user involvement in project definition.

- The company has provided timesharing access to a sophisticated data base management system (NOMAD) so that users can work rapidly with analysts to develop prototype information systems as an aid in requirements definition, and later to define revisions during maintenance and enhancement activities.
- Users also have timesharing access to a query system (RAMIS) to help design data structure and prototype screens during the system specification phase.

9. PROGRAMMING RESOURCES

- Consistent with its stage three status, the company is providing each analyst/programmer with interactive terminal access to a line-oriented time-sharing system.
- The company provides portable terminals for home use by selected programmers both for better trouble-shooting and for high-level (MACRO) program development.
- The PDS provides a training course that includes a programmed instruction course, IMS basics, and on-line "Help" files.
- Top professional personnel are used for creating macros to handle data base access, structured programming constructs, control blocks, IMS message interfaces and data structures.
- Company experience indicates that an entry-level programmer spends the first two weeks with an on-line terminal using the PDS course; within one month is doing some production work; and within two months is integrated into a programming group.

10. PRODUCTIVITY TOOLS

- Development of its in-house program development system (PDS) has moved the company solidly into stage three, "Efficiency".

- The PDS provides integrated and comprehensive automated support, from program design through maintenance, in the application development life cycle. The PDS fosters detailed systems definition and documentation as prerequisites for input to the PDS.
- The PDS automates and integrates the following features:
 - Uniform coding scheme for data, programs and systems.
 - Standardized integrated documentation.
 - Comprehensive data dictionary.
 - Timesharing user interface.
 - Very-high-level, structured procedural language.
 - Project planning and control.
 - Interactive self-training.
 - Multilevel testing.
 - Change and release control.
- The company's objective in building the PDS was to improve analyst/programmer productivity through standardization by automating as many of the program development and maintenance phases as possible.
- The company has realized the following benefits through using the PDS:
 - Redundancies are largely eliminated.

- Project control and productivity measurement statistics are automatically captured.
 - Up-to-date documentation is always available.
 - Automated generation of declarations eliminates inconsistencies.
 - Automated initialization of all tables eliminates one of the most common "bugs."
 - Generation of all interface control blocks reduces the need for high technical skills.
 - Maintenance of the data dictionary is automatically provided.
 - Production and maintenance of a global cross-reference is facilitated.
 - The programmer is allowed to concentrate on problem solution, as opposed to being concerned with the host system environment.
 - Automatic multilevel testing is standard.
 - Automatic change and release control is enforced.
 - Professional productivity is improved.
 - Maintenance requirements are reduced.
- The company is planning to augment the PDS with additional support to system design through a formal design methodology such as Jackson or Warnier-Orr.

II. INVESTMENT PAYOFF

- Although the company is continuing to invest in productivity improvements, it is beginning to experience payoff in various stages of the application development life cycle.
- The company has been able to reduce its maintenance effort from more than 50% of total life cycle costs to less than 30%. The company expects by 1985 to reduce professional personnel-related maintenance costs to a practical limit of 20% of the total life cycle costs.
 - A credit application that formerly required 20 professionals for maintenance, now (under the PDS) requires only three.
- The company is experiencing program development statistics averaging 2,000 lines of debugged source code per month for project team professionals.
- The company believes it has done about all it can do to aid productivity during the program development and maintenance process. It believes that its next breakthrough is coming at the front end during the application specification process. Specific payoff is expected from:
 - Placing 7-10% of professional staff directly with end users.
 - Implementation of program design techniques involving both end users and computer services professionals, jointly.
 - Use of query and other languages to develop prototypes as part of the definition phase, and later as part of the application enhancement phase.

D. CASE HISTORY #4 - UNSUCCESSFUL STAGE ZERO TRANSPORTATION COMPANY

I. THE COMPANY

- A Fortune 100 transportation company with revenues approaching \$300 million, this company is primarily engaged in intercoastal warehousing and transportation.
- Taking advantage of the rapid rise in truck and rail transportation rates resulting from escalating oil prices, the company has increased revenues exceeding 100% AAGR over the last five years.
- With more than 3,000 employees, the company is having its growing pains in:
 - Hiring qualified employees.
 - Establishing management control of its diversified operating locations.
 - Building an effective management information system.

2. EDP ORGANIZATION

- The company is having difficulty moving toward stage one, "Control," along with problems within the company as a whole, resulting from accelerated growth.
- An early attempt to centralize EDP at corporate headquarters and to establish an on-line information system for port operations failed when a large-scale Honeywell system could not operate the communications network as planned.
- The company was forced to pull back its plan, allowing operating divisions to use local EDP/computer services while it converted to an IBM 370 system.

- The corporate EDP facility now uses a large 370/158 and has a staff exceeding 100, with 45 professional analyst/programmers.
- Major attention is still being given to "cleaning up" after conversion of baseline accounting applications.
- The EDP manager, who came up through the ranks, has recently been replaced by a consultant while the company searches for a MIS director.

3. APPLICATION TYPE

- The company is completing conversion of its baseline accounting systems to operate in a batch environment on the corporate IBM system 370/158.
- The company is attempting to straddle stages one and two by implementing CICS for data entry from its operating centers, reporting to its EDP center at corporate headquarters.
 - A 3270 interface will be used to handle corporate data, relying on the local data processor to handle individual operations center requirements.
- The company plans to use IMS together with a data dictionary to implement integrated order entry, transfer and shipment applications across operating centers within the near future (1-2 yrs.).

4. EDP MANAGEMENT DESCRIPTION

- Corporate management is still focusing its attention on company revenue growth opportunities, and has not yet recognized the importance of information systems for assuring corresponding earnings growth.

- Problems associated with rapid growth have recently highlighted the need for management to focus attention on the inadequacy of current data systems performance.
- Inadequate to the task, the current EDP manager (an employee of the company from unit record days) has been replaced by a consultant while the company searches for a qualified EDP executive to become MIS director.
- Morale is low at the operating level, and the company is experiencing a 40% turnover rate among its EDP professional staff.
- Recent cancellation of the first project using a formal system development methodology resulted in dissolution of the project team, with most of the team members leaving the company.

5. CONTROL MECHANISM

- The company has discarded its current project management and control system for being too cumbersome and nonresponsive.
- The problems encountered in conversion and in establishing the CICS network have precluded scheduling new applications for implementation.
- In an effort to demonstrate system development methodology effectiveness, the EDP manager authorized establishment of a six-man project group in EDP to use PRIDE/Logik to define an on-line shipment control system.
- Operating without management mandate or user support, the group was six months into the design phase when user dissatisfaction forced removal of the EDP manager, cancellation of the project and transfer of the project responsibility (through consultant recommendation) to the user at the cognizant operations center.

- In an effort to give guidance to MIS development, the company recently established an EDP steering committee, consisting of operating corporate executives, including the president. The committee retained an executive consultant to give guidance and recommendations as to future company direction.

6. PERFORMANCE CHARACTERISTICS

- Since more than 80% of the analyst/programmer staff is assigned to converting and maintaining existing systems, little in the way of new systems are currently under development.
- The termination of the project that was to use PRIDE/ASDM to develop a comprehensive data dictionary and new on-line carrier system, leaves the company still struggling for an approach to move toward stage one, "Control."
- Still operating on an ad hoc basis and primarily concerned with getting the baseline accounting functions to work company-wide, the company has not yet focused its attention on improving performance through controlled systems definition and development.
- Backlog, currently extended in excess of 20 person years of professional effort, continues to grow.

7. QUALITY CONTROL

- In stage zero, "Chaos," the company has focused little attention on quality assurance.
- Lacking top-management support, both users and other EDP professionals resisted the concepts of specific test plans and incorporating quality assurance into systems definition using PRIDE/ASDM.

8. USER INVOLVEMENT

- With its widely dispersed coastal port operations, the company is a good candidate for distributed processing.
- Struggling first to establish control through centralization, and spurred by rapid company growth, corporate EDP is in strong conflict with user demands.
- Users claimed that they saw nothing but delay in implementing an on-line order entry and shipment control system using PRIDE/ASDM.
- In retrospect, the development team believes that it should have used other aids in addition to PRIDE/ASDM during design and specification.
 - User support would have benefited from the creation of prototype screens to "picture" for the end user what the system could be like.
 - By using prototypes, management could feel more comfortable that something was happening to offset the increased development time.

9. PROGRAMMING RESOURCES

- Although the company still accomplishes program development using batch techniques (stage zero), it is moving toward stage one.
- The company has ordered some 100 terminals, 60 for company operations using IMS, and 40 for on-line programming development under CMS.
- The development group was not able to successfully implement an automated data dictionary because program development was assigned a low priority when it came to allocating computer resources.
- The company's plans to implement IMS together with an automated data dictionary have been slowed by lack of assured access to computer resources.

Restrictions on data development have resulted in users' referring to the data base administrator as a "data deity."

10. PRODUCTIVITY TOOLS

- Though still at stage zero, the company is beginning to recognize the importance of computer-aided tools to assist in program development.
- The company is planning to use CMS as a means to begin supporting interactive program development.
- Transfer of new application definition responsibility to the cognizant operations centers, as recommended by the consultant, will still require on-line systems to support applications development.

11. INVESTMENT PAYOFF

- The company continues to pour money into EDP operations as productivity declines, with the widening gap between cost and benefit.
- Management has realized that something is wrong, has hired an EDP consultant, and is searching for a new EDP director.
- Its initial failure at using a systems development methodology will set the company back even more as it gropes for future direction.
- A step the company appears ready to take is to raise the importance of EDP in the corporate organization. Coincident with this decision, management must put teeth in its new EDP steering committee, giving the committee the charter to conduct strategic planning of MIS throughout the company, to establish firm priorities, to approve budgets and schedules, and to review results.

E. CASE HISTORY #5: OPTIMIZER III, MARK IV

I. THE COMPANY

- Fortune 1000 electronics manufacturer.
- Sales - \$250 million.
- Employees - 7,000.

2. EDP ENVIRONMENT

- 370/158; 7 megabyte memory; 90 terminals; running MVS, TSO, IMS.
- Also DEC PDP 20, used for engineering on a timesharing basis.
- Professional staff.
 - Fifteen percent maintenance (less than one person month of effort).
 - Forty-three percent changes and deletions.
 - Forty-two percent new system development.

3. PRODUCTIVITY PROBLEMS/SELECTED TECHNIQUES

- Inability to respond quickly to ad hoc user requests:
 - Mark IV.
- Inadequate response time (under TSO) for program development:
 - Optimizer III plus more hardware (memory and channels).

- Inability to do complete and accurate testing, especially of transactions.
 - In-house system to build data base of test data.

- Poor morale.

- Programmer frustration.

4. TECHNIQUE EVALUATION

- See Exhibit VI-1, describing the use of Optimizer III, and Exhibit VI-2, describing the use of Mark IV.

5. DEGREE OF SUCCESS

- Fair to good.
- Has reduced annual program maintenance and lowered budget from eight to six person years.
- Over the past three years, has increased number of operating transactions from 85,000 to 250,000 per month.
- Reliability of production runs has increased from 94% to 97% for all batch runs.
 - Reliability = number of jobs divided by number of abends.

6. COSTS

- Between \$150,000-200,000 for both.

EXHIBIT VI-1

COMPANY 5, PRODUCTIVITY TECHNIQUE EVALUATION: OPTIMIZER III

FACTOR	RESPONDENT'S COMMENTS
TRAINING REQUIRED	<ul style="list-style-type: none"> ● MINIMAL
RANGE OF BENEFITS	<ul style="list-style-type: none"> ● REDUCE CPU TIME ● ABILITY TO ANALYZE AND HIGHLIGHT INEFFICIENT CODE ● TEST ALL LOGIC PATHS ARE USED ● PRODUCE SMALLER LOAD MODULES
BENEFITS ACHIEVED	<ul style="list-style-type: none"> ● 7-10% REDUCTION IN CPU TIME ● INCREASED CODE EFFICIENCY
TIME TO ACHIEVE BENEFITS	<ul style="list-style-type: none"> ● IMMEDIATE FOR REDUCTION IN CPU TIME ● 1-2 YEARS FOR CODE EFFICIENCY
ABILITY TO MEASURE BENEFITS	<ul style="list-style-type: none"> ● VERY EASY ● CHARGE-BACK SYSTEM IS BASED ON CPU STATISTICS
JUSTIFICATION FOR ACQUISITION	<ul style="list-style-type: none"> ● PRESERVE, CONSERVE RESOURCES ● USE LESS MACHINE TIME ● USE LESS MEMORY
BASIS FOR SELECTION	<ul style="list-style-type: none"> ● ACTUAL DOLLAR SAVINGS
IMPLEMENTATION EXPERIENCE	<ul style="list-style-type: none"> ● EXCELLENT
IMPROVEMENTS NEEDED	<ul style="list-style-type: none"> ● BETTER TRAINING IN USE OF OPTIONS
INTERACTION WITH OTHER AIDS	<ul style="list-style-type: none"> ● NOT APPLICABLE
ACCEPTABILITY TO STAFF	<ul style="list-style-type: none"> ● EXCELLENT
ACCEPTABILITY TO MANAGEMENT	<ul style="list-style-type: none"> ● EXCELLENT

EXHIBIT VI-2

COMPANY 5, PRODUCTIVITY TECHNIQUE EVALUATION: MARK IV

FACTOR	RESPONDENT'S COMMENTS
TRAINING REQUIRED	<ul style="list-style-type: none"> ● MINIMAL FOR USE OF BASIC CAPABILITIES ● EXTENSIVE FOR USE OF FULL CAPABILITIES
RANGE OF BENEFITS	<ul style="list-style-type: none"> ● REDUCE DEVELOPMENT COSTS ● INCREASE END USER INVOLVEMENT ● RELIEVE PRESSURE ON EDP; ESPECIALLY FOR <u>AD HOC</u> REPORTS
BENEFITS ACHIEVED	<ul style="list-style-type: none"> ● REDUCE BACKLOG OF REQUESTS ● SAVINGS OF ONE TO TWO MAN YEARS ANNUALLY
TIME TO ACHIEVE BENEFITS	<ul style="list-style-type: none"> ● THREE TO SIX MONTHS
ABILITY TO MEASURE BENEFITS	<ul style="list-style-type: none"> ● FAIRLY NEBULOUS ● END USER SATISFACTION - LESS COMPLAINTS
JUSTIFICATION FOR ACQUISITION	<ul style="list-style-type: none"> ● REDUCE DEVELOPMENT COSTS ● MORE <u>AD HOC</u> REQUEST CAPABILITY TO END USERS
BASIS FOR SELECTION	<ul style="list-style-type: none"> ● WIDE INDUSTRY USE ● IMS INTERFACE
IMPLEMENTATION EXPERIENCE	<ul style="list-style-type: none"> ● INITIAL RESISTANCE FROM EDP ● GOOD WITH USERS
IMPROVEMENTS NEEDED	<ul style="list-style-type: none"> ● GREATER OPERATING EFFICIENCY ● GREATER FLEXIBILITY
INTERACTION WITH OTHER AIDS	<ul style="list-style-type: none"> ● GOOD INTERFACE WITH IMS
ACCEPTABILITY TO STAFF	<ul style="list-style-type: none"> ● NOW ACCEPTABLE TO EDP - USERS LIKE IT FOR "QUICK AND DIRTY" REPORTS ● "POLITICALLY EXCELLENT"
ACCEPTABILITY TO MANAGEMENT	<ul style="list-style-type: none"> ● DATA PROCESSING MANAGEMENT - ACHIEVED OBJECTIVES OF GREATER END USER SELF-HELP. ● END USER MANAGEMENT - EXCELLENT

7. BENEFIT DETERMINATION

- Intangible:
 - End users' use of Mark IV has greatly alleviated "political pressure" on DP.
 - There is greater end user satisfaction.
- Tangible:
 - Has fostered development of real measures of productivity.

8. COMMENTS IN RETROSPECT

- "Would have taken advantage of additional hardware much earlier."

9. FUTURE PLANS

- Automated documentation.
- Design standards.
- Standard modules.
- Involve quality assurance people in design effort.
- Initiate program for paraprofessionals to do maintenance tasks.
 - Housekeeping.
 - JCL, table changes, recompilations.
- Create code generators.

- Implement interactive debugging.
- Add transactions to test data base.

F. CASE HISTORY #6: WANG VS

1. THE COMPANY

- Fortune 500 office automation manufacturer.
 - Sales - \$600 million.
 - Employees - 11,000.

2. EDP ENVIRONMENT

- Shifting from IBM 370 to Wang VS systems.
- Currently running in dual environment.
 - IBM 3033 used primarily for production.
 - IMS, TSO, WYLBUR.
 - Some development.
 - Ten Wang VS systems used primarily for development.
 - Some new production.
- One hundred fifty people in MIS organization.

- Normal implementation mode is:
 - Development on Wang VS.
 - After sign-off by user, recompile and move into production on IBM 3033.
- Transfer of new applications into production mode being very successfully done by nonprofessional personnel moving into EDP.
- Most of the documentation being done by nonprofessional personnel, with heavy use of word processing.
- Very nonbureaucratic atmosphere.
 - People willing to help others.
 - Many young, bright people.

3. PRODUCTIVITY PROBLEMS/SELECTED TECHNIQUES

- Response time using TSO/IMS was severely limiting new development:
 - Switch to Wang VS for application development.
 - Use of Wang VS user aid package.
- Incentive to establish "beta test" site for Wang VS:
 - Demonstrate increased productivity of professionals.

4. TECHNIQUE EVALUATION

- See Exhibit VI-3 for description of Wang VS.

EXHIBIT VI-3

COMPANY 6, PRODUCTIVITY TECHNIQUE EVALUATION: WANG VS

FACTOR	RESPONDENT'S COMMENTS
TRAINING REQUIRED	<ul style="list-style-type: none"> ● ONE WEEK
RANGE OF BENEFITS	<ul style="list-style-type: none"> ● EXCELLENT USER FRIENDLINESS ● RAPID ON-LINE INTERACTION ● COMPREHENSIVE MENU-DRIVEN SELF-TEACHING SYSTEM
BENEFITS ACHIEVED	<ul style="list-style-type: none"> ● 5-10X IMPROVED PRODUCTIVITY OVER TSO ● EASIER THAN TSO TO LEARN AND USE ● PROVIDE MORE RESOURCES TO EACH PROGRAMMER
TIME TO ACHIEVE BENEFITS	<ul style="list-style-type: none"> ● ABOUT ONE YEAR
ABILITY TO MEASURE BENEFITS	<ul style="list-style-type: none"> ● LOWER TURNOVER ● HIGHER MORALE ● IMPROVED OUTPUT
JUSTIFICATION FOR ACQUISITION	<ul style="list-style-type: none"> ● DOLLAR SAVINGS USING WANG HARDWARE
BASIS FOR SELECTION	<ul style="list-style-type: none"> ● COST/BENEFIT ANALYSIS
IMPLEMENTATION EXPERIENCE	<ul style="list-style-type: none"> ● INITIALLY REQUIRED LOTS OF SUPPORT AND "FIXES" ● RELIABILITY PROBLEMS WITH NEW HARDWARE ● NOW EVERYTHING IS WORKING FINE
IMPROVEMENTS NEEDED	<ul style="list-style-type: none"> ● PL/1 COMPILER ● NETWORKING
INTERACTION WITH OTHER AIDS	<ul style="list-style-type: none"> ● SOME PROBLEMS OPERATING WITH TWO ENVIRONMENTS - IBM 3033 TSO/IMS AND WANG VS.
ACCEPTABILITY TO STAFF	<ul style="list-style-type: none"> ● VERY HIGH
ACCEPTABILITY TO MANAGEMENT	<ul style="list-style-type: none"> ● EXCELLENT

5. DEGREE OF SUCCESS

- Very successful; have greatly reduced dependency on specific people with specific knowledge (i.e., IMS).
- Systems people have become more involved with applications development functions.
- Greatly reduced frustration through "user friendliness" of system.
- Improved program development productivity by a factor of 10 to 15.
- System designed from the start to operate in an on-line environment.
- System completely menu-driven, relieving user from memorizing all the command codes, etc., to use system effectively.
- Response time far superior to TSO/WYLBUR.

6. COSTS

- While detail split is not available, total EDP costs are less than before switchover.

7. BENEFIT DETERMINATION

- Benefits:
 - Increased productivity (10-15X).
 - Professional turnover now minimal.
 - System totally available - one terminal for each analyst/programmer.

- Menu-driven, easy to learn, "user-friendly," on-line environment.
- Full complement of utilities and catalogue of "skeleton code."

- Drawbacks:

- Had to switch from PL/I to COBOL.
- Analyst/programmer reticence to switch from IBM mainstream.
- Initial lack of full complement of software aids.
- Limit of 32 users per Wang VS system.

- Measurement:

- Only on a gross basis.
- Total EDP costs less, and doing more work.
- Users more satisfied.
- Now developing specific measurements.

8. COMMENTS IN RETROSPECT

- Don't know a way to have done it differently.

9. FUTURE PLANS

- Reduce 3033 development, then try to switch completely to Wang systems.
- Replace Wang 2200/VS systems with Wang VS/100s.

- Improve communication capabilities between systems:
 - IBM and Wang.
 - Wang and Wang.
- Emphasize a "transactional" philosophy rather than a "data base" approach to applications development.
- Provide more end user facilities and capabilities.
- Incorporate electronic mail into system development.

G. CASE HISTORY #7: SADT

I. THE COMPANY

- A Fortune 50 bank.
 - Assets - \$40 billion.
 - Employees - 17,500.

2. EDP ENVIRONMENT

- Centralized data processing using MVS, TSO, IMS.
- Two 3033s, 1-3032, 2-370s, multiple HP minis.
- One 3033 dedicated to development and testing.
 - Thirty percent new development.

- Seventy percent maintenance.

3. PRODUCTIVITY PROBLEMS/SELECTED TECHNIQUES

- Basis for selection:

- To see how data processing could improve its services to the company.
- To determine what new or improved financial services the company could offer.

- Techniques utilized:

- Company's five-year plan focused attention on "productivity of data systems development."
- Four major areas selected were:
 - . Interactive program development.
 - Within data processing - TSO.
 - For end users - "user-friendly" languages and data accessibility.
 - . Structured analysis, design and programming.
 - SADT.
 - Warnier-Orr.
 - . Standardized testing methodology.
 - Trailblazer.

- . Development of productivity measurements.
 - Not lines of code.
 - Better cost/benefit analysis.
 - Better estimating and pricing.
 - Charge-back system understandable by users.
- Other areas considered were:
 - . Bring life cycle to "minis."
 - . Improve site environment and security.
 - . Improve management of application "changes."
 - Release control.
 - Decide when to redo system.
 - . Develop concept of prototyping or "throw-away" software.

4. TECHNIQUE EVALUATION

- See Exhibit VI-4 for a description of SADT.

5. DEGREE OF SUCCESS

- Interactive programming:
 - Average to date.

EXHIBIT VI-4

COMPANY 7, PRODUCTIVITY TECHNIQUE EVALUATION: SADT

FACTOR	RESPONDENT'S COMMENTS
TRAINING REQUIRED	<ul style="list-style-type: none"> ● ONE-WEEK CLASS FOR EACH OF THREE PHASES ● TO BECOME PROFICIENT, NEED CONSULTATION HELP ● LEARNING CURVE IS SIX MONTHS AND MULTIPLE PROJECTS
RANGE OF BENEFITS	<ul style="list-style-type: none"> ● ENHANCED END USER INTERACTION AND SATISFACTION ● BREAKS LIFE CYCLE INTO LOGICAL WORK TASKS ● INCLUDES "OUTSIDERS" IN COMPREHENSIVE TECHNICAL REVIEW PROCESS
BENEFITS ACHIEVED	<ul style="list-style-type: none"> ● TOO SOON TO TELL
TIME TO ACHIEVE BENEFITS	<ul style="list-style-type: none"> ● SEEMS TO TAKE LONGER IN DESIGN PHASE ● SEEMS TO SHORTEN IMPLEMENTATION PHASE ● END PRODUCT HAS FEWER ERRORS
ABILITY TO MEASURE BENEFITS	<ul style="list-style-type: none"> ● BASICALLY MANUAL METHODS ● VERY PEOPLE-DEPENDENT ● VERY DIFFICULT AS YET TO MEASURE MAINTENANCE BENEFITS
JUSTIFICATION FOR ACQUISITION	<ul style="list-style-type: none"> ● "THINGS WERE A MESS" ● NEEDED TO STANDARDIZE DEFINITION EFFORT, CATCH ERRORS EARLY, REDUCE MAINTENANCE AND IMPROVE DOCUMENTATION
BASIS FOR SELECTION	<ul style="list-style-type: none"> ● COMPETITIVE ANALYSIS ● CREDIBILITY OF VENDOR ● ADDITIONAL CONSULTATION AVAILABLE
IMPLEMENTATION EXPERIENCE	<ul style="list-style-type: none"> ● STILL IN FIRST STAGES WITH NO PROJECT YET COMPLETED ● HAVE NOT YET HAD ANY INDICATIONS THAT SOMETHING IS WRONG
IMPROVEMENTS NEEDED	<ul style="list-style-type: none"> ● TOO SOON TO TELL ● PERHAPS BETTER TRAINING
INTERACTION WITH OTHER AIDS	<ul style="list-style-type: none"> ● HAVE USED WITH WARNIER-ORR FOR MODULE NAMES, CHRONOLOGICAL REQUIREMENTS AND DATA DECOMPOSITION
ACCEPTABILITY TO STAFF	<ul style="list-style-type: none"> ● EXPERIENCED "STRUCTURE SHOCK" ● RESISTANCE BY EXPERIENCED PEOPLE ● FEAR OF ADDITIONAL OVERHEAD
ACCEPTABILITY TO MANAGEMENT	<ul style="list-style-type: none"> ● INITIAL CONCERN ABOUT SCHEDULE AND "WHEN DO WE CODE?" ● EXPECT ATTITUDES WILL CHANGE WITH FIRST SUCCESSES

- Three-and-one-half programmers/terminal.
- Structured analysis:
 - Excellent.
- Standard testing methodology:
 - Too early to tell.
- Productivity measurements:
 - Slow.
 - Have identified life cycle deliverables.

6. COSTS

- Interactive programming:
 - Not yet measured.
- Structured analysis:
 - \$250 thousand to date over 2.5 years.
- Standard testing methodology:
 - \$300 thousand to date over three years.
- Productivity measurements:
 - Minimal to date.

7. BENEFIT DETERMINATION

- Benefits:

- "Have changed the way we do business."
- Standard type of work packages.
- Early, high visibility of documentation.
- Greater and earlier end user involvement.
- Improved communication.
- Now developing standards.
- Now beginning to gather statistics.

- Drawbacks:

- Initial ripple of misunderstanding from people threatened by "productivity" measurement.
- Problems of introducing change into an organization are of a sensitive nature.

- Measurements:

- No hard numbers.
- People in data processing beginning to realize information is a "business."
- End users are more satisfied, especially with data accessibility.

8. COMMENTS IN RETROSPECT

- "Would pay more attention to turnover, people psychology, environment."
- "Would construct initial team on pilot effort with more analysts rather than programming professionals."

9. FUTURE PLANS

- More computer-aided program development support.
- More computer-aided application analysis.
- Tailor aids to environment.
- Become more independent of hardware vendor.
- Integrate more aids.

H. CASE HISTORY #8: DATA CATALOG II

I. THE COMPANY

- A middle-market consumer products retailing company.
 - Annual revenues - \$100-200 million.
 - Employees - 1,000-2,000.

2. EDP ENVIRONMENT

- - 370/135; one-half megabyte.

- Using DOS/VS, Power.
 - Ten systems analysts and programmers.
3. PRODUCTIVITY PROBLEMS/SELECTED TECHNIQUES
- Are in third year of four-year consolidation and standardization plan.
 - Buy/acquire packages whenever possible.
 - Most packages are worth the investment.
 - Trade computer time for packages where attractive.
 - Selected data dictionary as the best means to foster consolidation and standardization.
 - Programs, documentation, JCL, etc., catalogued in addition to data.
 - Control movement into data base environment.
 - Control use of structured design and development.
 - Get users involved and committed at the front end through use of the data dictionary.
 - Ensure that documentation precedes programming, through use of the data dictionary.
4. TECHNIQUE EVALUATION
- See Exhibit VI-5 for a description of Data Catalog II.

EXHIBIT VI-5

COMPANY 8, PRODUCTIVITY TECHNIQUE EVALUATION: DATA CATALOG II

FACTOR	RESPONDENT'S COMMENTS
TRAINING REQUIRED	<ul style="list-style-type: none"> ● PRIMARILY ON-THE-JOB TRAINING: TWO-DAY TRAINING COURSE ● MOST DIFFICULTIES ARE IN FITTING IT IN THROUGH-OUT LIFE CYCLE ● LEARNING CURVE IS TWO MONTHS
RANGE OF BENEFITS	<ul style="list-style-type: none"> ● ALL THE WAY THROUGH LIFE CYCLE
BENEFITS ACHIEVED	<ul style="list-style-type: none"> ● HAS BEEN INTEGRATED INTO MOST PHASES OF LIFE CYCLE
TIME TO ACHIEVE BENEFITS	<ul style="list-style-type: none"> ● ONE TO TWO YEARS
ABILITY TO MEASURE BENEFITS	<ul style="list-style-type: none"> ● VERY DIFFICULT, MOSTLY IN COST AVOIDANCE AND REDUCING "DISASTERS"
JUSTIFICATION FOR ACQUISITION	<ul style="list-style-type: none"> ● ALLOW GREATER FLEXIBILITY IN HANDLING CHANGES ● GREATER PAYOFF AS SYSTEMS BECOME MORE COMPLEX ● SAVINGS MOUNT AS 50% OF DATA ELEMENTS, ETC., ARE IN DICTIONARY
BASIS FOR SELECTION	<ul style="list-style-type: none"> ● STRUCK DEAL IN EXCHANGE FOR MACHINE TIME
IMPLEMENTATION EXPERIENCE	<ul style="list-style-type: none"> ● GENERALLY GOOD
IMPROVEMENTS NEEDED	<ul style="list-style-type: none"> ● ON-LINE FOR DOS/CICS ● MORE TERMINAL CAPABILITY INCLUDING GRAPHICS
INTERACTION WITH OTHER AIDS	<ul style="list-style-type: none"> ● GENERIC TOOL FOR LIFE CYCLE
ACCEPTABILITY TO STAFF	<ul style="list-style-type: none"> ● FINE
ACCEPTABILITY TO MANAGEMENT	<ul style="list-style-type: none"> ● SUPERB ● IMPACT ON SCHEDULE HAS BEEN VERY POSITIVE ● EXCELLENT DATA CONTROL

5. DEGREE OF SUCCESS

- Very successful.
- Can now introduce structured design and development by running everything through the data dictionary.
- The dictionary is used by all users and MIS professionals for all projects.

6. COSTS

- Traded excess computer time for rights to use package.
- The use of the data dictionary was essentially free.

7. BENEFIT DETERMINATION

- Benefits:
 - Dictionary controls and supports all areas for:
 - Report content and distribution.
 - Scheduling.
 - Documentation.
 - Can use "what if" types of searches to determine:
 - What procedures are affected.
 - What programs are affected.
 - What files, data bases, etc., are affected.

- Putting all COBOL source code into dictionary.
- Drawbacks:
 - Currently have no on-line capability for DOS/CICS.
- Measurement:
 - Tangible:
 - Have not yet tried to measure tangible dollar costs.
 - Intangible:
 - Standardization; centralized control of data, procedures, etc.
 - Separate "people characteristics" from systems/programs.

8. COMMENTS IN RETROSPECT

- "Would go on-line immediately."
- "Would make documentation easier and faster to accomplish."
- "If people are given a tool that is perceived to be easier and faster to use, they will turn to use it."
- "Mistaken notion that EDP people are resistant to change. Rapid change is byword of MIS environment."
 - "But a finite limit on how much change data processing can handle simultaneously."

9. FUTURE PLANS

- Go on-line as soon as possible.
- Upgrade system.
 - Double memory.
 - Triple number of terminals.
 - Qaudruple processor power.

I. CASE HISTORY #9: PSL/PSA

I. THE COMPANY

- Subsidiary of public utility.
 - Revenues - \$2 billion.
 - Employees - 41,000.

2. EDP ENVIRONMENT

- Multiple IBM 3033s.
- RJE to parent for computer-aided system specification and design.

3. PRODUCTIVITY PROBLEMS/SELECTED TECHNIQUES

- Basis for selection:

- Parent recommended its use.
 - Had been used on a pilot basis before.
 - Wanted better documentation in definition phase.
 - Greater control over requirements determination.
 - Greater communications/standardization among personnel on large projects.
 - Wanted all changes controlled through librarian.
 - Greater user involvement in definition phase.
 - Clean interface and good documentation between definition group and implementation group.
- Techniques utilized:
 - Use PSL as a documentation tool, from management point of view.
 - Do not yet use PSL for design. Do not use:
 - Procedure division.
 - Analysis capabilities to any great extent.
 - Use two basic data bases for a project.
 - System-related information; data and structure.
 - Administrative information, memos, reports.

- Also use:
 - . Data dictionary.
 - . Structured systems analyses (Gane and Sarson).

4. TECHNIQUE EVALUATION

- See Exhibit VI-6 for a description of PSL/PSA.

5. DEGREE OF SUCCESS

- The jury is still out.
- Varies greatly with different people.
 - Documentation good across the board.
 - Structure very difficult for end users to grasp.
- Input to the data base dictates the usefulness of analysis phase.
 - How much to put in.
 - How detailed it is.

6. COSTS

- Have not yet tracked costs.

7. BENEFIT DETERMINATION

- Benefits:

EXHIBIT VI-6

COMPANY 9, PRODUCTIVITY TECHNIQUE EVALUATION:

PSL/PSA

FACTOR	RESPONDENT'S COMMENTS
TRAINING REQUIRED	<ul style="list-style-type: none"> ● ALL EDP AND USER PERSONNEL MUST BE TRAINED ● FIVE-DAY FORMAL CLASS ● LEARNING CURVE ONE WEEK FOR LIBRARIANS, TWO TO THREE MONTHS FOR SYSTEMS DEFINERS
RANGE OF BENEFITS	<ul style="list-style-type: none"> ● PROVIDES OUTSTANDING DOCUMENTATION BETWEEN PROJECT TEAM AND USERS ● DOCUMENTATION AT FRONT END AIDS IMPLEMENTATION & MAINTENANCE ● CAN SEE SYSTEMS STRUCTURE, DATA HIERARCHY
BENEFITS ACHIEVED	<ul style="list-style-type: none"> ● CAN'T TELL YET OVERALL ● EXCELLENT RESULTS IN UNCOVERING REDUNDANCIES AND WHAT'S MISSING ● COMMUNICATION QUALITY IMPROVED
TIME TO ACHIEVE BENEFITS	<ul style="list-style-type: none"> ● BASICALLY DEPENDS ON SPEED OF INPUT
ABILITY TO MEASURE BENEFITS	<ul style="list-style-type: none"> ● TOO SOON TO TELL ● NOTHING TO MEASURE AGAINST
JUSTIFICATION FOR ACQUISITION	<ul style="list-style-type: none"> ● PREVIOUS PARENT COMPANY DECISION AND PILOT EXPERIENCE
BASIS FOR SELECTION	<ul style="list-style-type: none"> ● IMPROVED DOCUMENTATION AND COMMUNICATION ● EXPECTATION THAT OVERHEAD AT FRONT END WILL BE MORE THAN OFFSET DURING MAINTENANCE
IMPLEMENTATION EXPERIENCE	<ul style="list-style-type: none"> ● HARD TO GET OFF THE GROUND BUT OK NOW ● MAINTAINABILITY IS QUESTIONABLE
IMPROVEMENTS NEEDED	<ul style="list-style-type: none"> ● ON-LINE/INTERACTIVE DEFINITION INCLUDING INTERACTION WITH DATA DICTIONARY ● BETTER FORMS FOR EACH OBJECT TYPE ● ELIMINATE NAMING RESTRICTIONS
INTERACTION WITH OTHER AIDS	<ul style="list-style-type: none"> ● HAS BEEN INTERFACED WITH STRUCTURED SYSTEMS ANALYSIS; DATA OBJECTS MAP TO PSL OBJECTS; I/O MAPS TO PSL ENTITIES, ETC. ● HAS BEEN INTERFACED WITH A DATA DICTIONARY
ACCEPTABILITY TO STAFF	<ul style="list-style-type: none"> ● SEEMS TO WORK NICELY WITH DATA PROCESSING STAFF ● END USER LIKES BASIC DOCUMENTATION ASSISTANCE ● END USER STILL MYSTIFIED BY METHODS STRUCTURE
ACCEPTABILITY TO MANAGEMENT	<ul style="list-style-type: none"> ● ALTHOUGH NOT ALL CAPABILITIES ARE YET BEING USED, PSL/PSA IS PERCEIVED AS AN IMPROVEMENT, ESPECIALLY IN SYSTEM DOCUMENTATION

- Greater communication and understanding of large systems.
 - Can recognize missing specifications easier and earlier.
 - Structured reports are very useful.
 - . Hierarchy of data.
 - . Where data is used by processes.
 - . All outputs from a process, etc.
 - Drawbacks:
 - There is a large training overhead at the front end.
 - . All data processing and particularly user professionals involved must be trained.
 - Produces lots of paper.
 - Do not know if PSL data bases are truly maintainable.
 - Measurement:
 - Have not attempted to measure yet.
8. COMMENTS IN RETROSPECT
- "Perhaps PSL should be used only during definition phase and not throughout life cycle."
 - "Would have improved training courses and material before implementing."

- "Possibly would wait for on-line interactive capability including data dictionary."

- "Would train and use more librarians."

9. FUTURE PLANS

- Increase use of analysis experts.
- Use procedure section for design phase.
- Implement on-line/interactive capabilities.

J. CASE HISTORY #10: UNIVERSAL PRODUCTIVITY PACKAGE (UPP)

I. THE COMPANY

- Fortune 100 defense electronics conglomerate division.
 - Revenues - \$3.7 billion.
 - Employees - 67,000.

2. EDP ENVIRONMENT

- 370-158 using MVS, IMS, COBOL.
- One hundred forty staff members.
- One hundred programmers.
 - Routine maintenance - 5%.

- Maintenance changes and enhancements - 70%.
- New development - 25%.

3. PRODUCTIVITY PROBLEMS/SELECTED TECHNIQUES

- Basis for selection:

- Dollar rate per hour of programming professionals' time is rapidly escalating.
- Wanted long-range benefit in maintenance area.
- Wanted to put discipline in development process.
- Wanted to develop standards and consistency.

- Techniques selected:

- APL, ROSCOE, UCC10, DYL260.
- Reusable code.
- Program skeletons.

4. TECHNIQUE EVALUATION

- See Exhibit VI-7 for a description of UPP.

5. DEGREE OF SUCCESS

- New people grow very fast.
- Resistance fading as experienced people "get on board."

EXHIBIT VI-7

COMPANY 10, PRODUCTIVITY TECHNIQUE EVALUATION: UNIVERSAL PRODUCTIVITY PACKAGE (UPP)

FACTOR	RESPONDENT'S COMMENTS
TRAINING REQUIRED	<ul style="list-style-type: none"> ● FIVE-DAY COURSE FOR USERS, THREE-DAY COURSE FOR SUPERVISORS AND MANAGERS ● SHORT LEARNING CURVE, ONE TO TWO MONTHS OF USE ON THREE PROJECTS
RANGE OF BENEFITS	<ul style="list-style-type: none"> ● MORE EFFECTIVE USE OF PERSONNEL ● GREATER DISCIPLINE IN DEVELOPMENT PROCESS ● ESTABLISH SPECIFIC OBJECTIVES AND MEASURE AGAINST THEM
BENEFITS ACHIEVED	<ul style="list-style-type: none"> ● REDUCE PEOPLE COST BY 20% WITH LITTLE OR NO LOSS IN SERVICE ● BEGINNING PAYOFF IN MAINTENANCE AREA ● ACCOMPLISHING MORE ON THE FRONT END
TIME TO ACHIEVE BENEFITS	<ul style="list-style-type: none"> ● SOME BENEFITS ALMOST IMMEDIATELY ● ONE TO TWO YEARS THROUGHOUT ENTIRE ORGANIZATION
ABILITY TO MEASURE BENEFITS	<ul style="list-style-type: none"> ● OUTSTANDING MANAGEMENT REPORTING ● 60-70% OF NEW DEVELOPMENT CODE IS SKELETON OR REUSED FOR IMS APPLICATION ● 40% FOR NON-IMS APPLICATIONS
JUSTIFICATION FOR ACQUISITION	<ul style="list-style-type: none"> ● REDUCE COSTS ● INTRODUCE DISCIPLINE AND STANDARDIZATION INTO DEVELOPMENT PROCESS ● DEVELOP CONSISTENTLY HIGH QUALITY SYSTEMS
BASIS FOR SELECTION	<ul style="list-style-type: none"> ● DID COMPETITIVE ANALYSIS AND FOUND NOTHING EQUIVALENT AVAILABLE ● DECIDED ON IN-HOUSE DEVELOPMENT
IMPLEMENTATION EXPERIENCE	<ul style="list-style-type: none"> ● VERY GOOD ● WILL IMPROVE AS SPECIFIC OBJECTIVES ARE WRITTEN FOR THE USE OF SYSTEM
IMPROVEMENTS NEEDED	<ul style="list-style-type: none"> ● FORTRAN AND NONPROCEDURAL LANGUAGE ● REPORTING AND DATA COLLECTION SYSTEM IMPROVEMENTS ● INTERACTIVE DICTIONARY & CODE GENERATOR CAPABILITY
INTERACTION WITH OTHER AIDS	<ul style="list-style-type: none"> ● EXCELLENT WITH IMS ● GOOD WITH COBOL ANALYZER/STANDARDIZER
ACCEPTABILITY TO STAFF	<ul style="list-style-type: none"> ● RESISTANCE FROM EXPERIENCED STAFF AT FIRST ● WELL LIKED BY NEW HIRES
ACCEPTABILITY TO MANAGEMENT	<ul style="list-style-type: none"> ● EXCELLENT; ALLOWS SPECIFICATION OF GOALS, OBJECTIVES AND DIRECTION ● CONTROLS COSTS

- Able to develop specific objectives for program development.
 - Seventy-five to eighty-five percent of all new COBOL programs will use "skeletons."
 - Seventy percent of all new COBOL programs will use "reusable" code.

6. COSTS

- Initially about \$250 thousand.
- Continuing investment of about three person years.
- Staff responsible for:
 - Code certification.
 - Standard data names (dictionary).
 - Walk-through.
 - Evaluating new tools/aids.
 - Common reusable code.

7. BENEFIT DETERMINATION

- Benefits:
 - Twenty percent lower staffing than two years ago, with service to users as good or better.
 - Knowledge and awareness of advanced aids attractive in recruiting efforts.

- Drawbacks:

- Some senior people feel that:
 - . They are losing some of their job security.
 - . Their creativity is being inhibited.
- Fear of being judged on ability to use techniques.

- Measurement:

- Have developed control and reporting system to encourage/monitor use of skeleton and reusable code.

8. COMMENTS IN RETROSPECT

- "Would do a better job of marketing the techniques to staff professionals."
- "Would solicit more input from prospective users."
- "Would do a more carefully phased implementation."

9. FUTURE PLANS

- Enhance system to include:
 - FORTRAN.
 - Greater use of data dictionary.
 - Interactive capability with code generators.
 - Add nonprocedural language capability.

VII MEASUREMENT AND EVALUATION STRATEGIES

VII MEASUREMENT AND EVALUATION STRATEGIES

A. OBJECTIVES IN MEASUREMENT AND ASSESSMENT

- If there were no intention to improve, it would not be necessary to provide any kind of evaluation; but measurement may still be necessary for billing, planning or other common business purposes.
 - This distinction is frequently blurred in the minds of EDP managers, who tend to treat the subject of measurement as one-dimensional, boring, difficult and required but not useful.
 - Because of this confusion, EDP measurement and evaluation are frequently poorly performed, and give evidence of all of the undesirable characteristics just described.
 - The one area in which measurements have sometimes been more successful is computer operations; but even here there are many disappointing examples of how not to run a measurement program.
- This chapter attempts to differentiate between two kinds of evaluation methods:
 - Measurements, which are quantitative, rigorous and scientific (or at least purport to be so).

- Assessments, which may be quantitative, but which have far fewer pretensions regarding rigor and scientific method.
- As described below, both kinds of evaluation can provide feedback to increase productivity in the software development and implementation process.
- In order to select an appropriate set of techniques for a particular EDP organization, the following set of uses for assessment and measurement are defined.
 - There may be others that apply to the particular situation, which should be made explicit before choosing any specific techniques.
- The state of the art of measurement today does not support interorganizational comparisons very well. It is possible, however, to evaluate an organization against its own previous achievements and to work toward refining measurement techniques that will eventually support interorganizational comparisons.

B. USES OF MEASUREMENT AND ASSESSMENT METHODS

I. EVALUATION OF THE SOFTWARE PRODUCT

- What are the desirable characteristics of the software product? How can they be differentiated from less desirable characteristics, especially as determined from the output (quality, effectiveness of function) rather than from the input (time and cost to produce, maintain and operate)?
 - Many measurements focus on inputs (e.g., cost per 1,000 lines of code) rather than on such question as:
 - Is the code doing what was intended?

- . Is it reliable?
 - . Can it be modified easily?
 - . Were x-thousand lines needed in the first place?
- It is certainly not easy to answer these questions, but some techniques lend themselves better to evaluation of the software product, rather than the software process.
 - In turn, they may suggest new ways to produce software that result in a higher-quality product.
 - This has been the experience of the aerospace industry, where extreme requirements for reliability have forced the implementation of more effective design and certification practices.

2. IMPROVING THE DEVELOPMENT PROCESS

- Measurements can improve the software development process by assisting in:
 - Estimating development progress. ("Are we on schedule and within budget?")
 - Predicting resource requirements. (If 1,000 lines of code took four person months in project A, just completed, will it take the same amount of time to do upcoming project B, also estimated to involve 1,000 lines of code?)
 - Identifying problems in methodology or design. (If measurement could provide valid feedback in these areas, it could give a very powerful assist to the methodologies aimed at improving the development process, or conversely, it could eliminate from consideration those methodologies that are less successful.)

- Fulfilling the first objective (estimating progress) probably accounts for the bulk of measurement now going on.
 - While potentially useful, the objective has given much of measurement a bad name by strongly propelling participants toward certain answers ("We are on schedule") until it is too late to take corrective action.
 - The major difficulty lies not with the objective, but with the fact that the "measurement" is usually performed in a highly subjective manner.

3. IMPROVING THE MAINTENANCE PROCESS

- Maintenance now consumes a large proportion of the software dollar and often produces a mediocre product. A desirable objective is to produce measurements that will:
 - Indicate software maintenance quality, both process and product.
 - Indicate whether a maintained software product is stable.
 - Identify changes that have been made during the maintenance effort, and what their likely side effects will be.
- These areas represent frontiers in the measurement process.

4. ASSESSING USER SATISFACTION

- This kind of assessment is seemingly far less rigorous and scientific than such things as counting costs or bugs per 1,000 lines of code. However, a moment's reflection shows that these more "scientific" measures are really only surrogates for user satisfaction. ("Is management happy over how much the job costs?")

- In the final analysis, a bug is a bug if a user thinks so, and the cost is right if the user could not effectively do his or her job in a less expensive way.

5. ASSESSING ORGANIZATIONAL EFFECTIVENESS

- This objective is even further away from scientific measurement. However, it is probably the most important of all: the best tools are worthless if the workers are continually unhappy, are unable to use them effectively, or feel impelled to leave.

C. ASSESSMENT AS A MANAGEMENT TOOL

- Assessment is often overlooked as an evaluation tool for EDP managers. Assessment methods have been used for so long that they may no longer be fashionable: papers on assessment techniques are not often published in computer journals.
- Nevertheless, assessment has definite advantages.
 - It is easily implemented and unobtrusive.
 - It is often sufficiently accurate.
 - It can be a bridge to a more formal system.
 - It can provide nonquantifiable information that is not available any other way.
- On the other hand, assessment methods have shortcomings:
 - There is an inherent fuzziness to them.

- Even where superficially quantifiable (e.g., numeric weighting scales of satisfaction, performance, etc.), there are often insurmountable problems in the reproducibility, relativity (my "3" may be your "4") and stability of findings over time.
- At bottom, assessments are subjective, even if semi-objective reference points are included (e.g., "Diligence = 3, if an employee spends more than 10 minutes a day talking at the water fountain").
- Management, especially from scientific or engineering backgrounds, tend to distrust assessments. Consequently it is often hard to justify actions based on assessment findings.
- On balance, though, a number of areas are very well suited to assessment methods; e.g., determining:
 - User satisfaction.
 - Employee satisfaction and goals.
 - Employee and management performance and effectiveness (especially when assessment is used in conjunction with goal-setting methodologies such as management-by-objectives).
 - Employee skills (via external assessment as well as self-assessment).
 - Factors contributing to a project's success or failure.
 - Other "intangibles."
- These assessments can easily be done internally. They need not be complex or overly sophisticated. However, assessments should:

- Use written survey instruments (otherwise the assessment becomes a "How-are-you-doing" exercise).
 - Combine menu and scoring questions with open-ended questions.
 - Have the ground rules for assessment use known in advance. (Will the results be published? To what extent will respondent identities become widely known, etc.).
 - Provide immediate feedback to participants, often in the form of a joint review of their responses.
 - Instigate specific responses and/or identifiable changes as a result of the assessment.
- Above all, assessments must be taken seriously, and not become a public relations exercise or be performed because there is a vague belief that assessments are a good idea.

D. MEASUREMENT AS A MANAGEMENT TOOL

I. THE GENERAL PROBLEM

- The problem with assessment is that it is underrated; measurement, on the other hand, may be overrated in the EDP productivity arena.
- There are many dimensions to measurement. These dimensions are sometimes confused or misused.
 - Seemingly objective measurements may overlap or be based on subjective factors.

- There are many schools of thought on measurement. So far there are few signs of a comprehensive approach to unify the various partial concepts and practices.
- Among the many dimensions of measurement, each with its own set of contrasting attributes, are:
 - Absolute versus relative results.
 - Process versus product focuses.
 - Cost versus rate.
 - Development versus maintenance versus operation versus life cycle cost.
 - Individuals' versus systems' performances.
 - Evaluative versus predictive applications.
- Measurement is a quest for uniformity. What is desired is not just a single instance of uniformity, but a characteristic of measurement that, for example, can compare the cost of software and be assured of a uniform applicability:
 - Across industries.
 - Across company size.
 - Across languages.
 - Over time.
- Clearly, EDP has not yet produced such a measurement, and perhaps never will.

- Finally, measurement itself is difficult because, in producing software:
 - Exactly the same task is never done twice.
 - Measurement of professional design and programming activities is inherently complex.
 - There are a large number of uncontrolled variables to account for.
 - There is no agreement on the relative weights of critical factors within the software product (maintainability, reliability, etc.).
- As shown in Exhibit VII-1, this situation produces numerous managerial dilemmas, for example:
 - What should be counted? Pages? Lines? Bytes? There is no agreed-upon transformation between high-level and assembler languages to standardize line counts. Even within a high-level language, there are different conventions determining which kinds of lines to count.
 - In a maintenance environment like that shown in Exhibit VII-2, there are even more alternatives on what to count or not count, since many different "languages" are used side by side, with overlapping functions. Should only changes and additions be counted, or are deletions equally important? Should the same change, made in many places, be counted only once or each time it is made?
- Arguments supporting measurement are that it:
 - Provides a foundation for comparison.
 - Conveys seriousness of purpose.
 - Facilitates planning.

EXHIBIT VII-1

THE MEASUREMENT DILEMMA: PAGES, LINES, BYTES

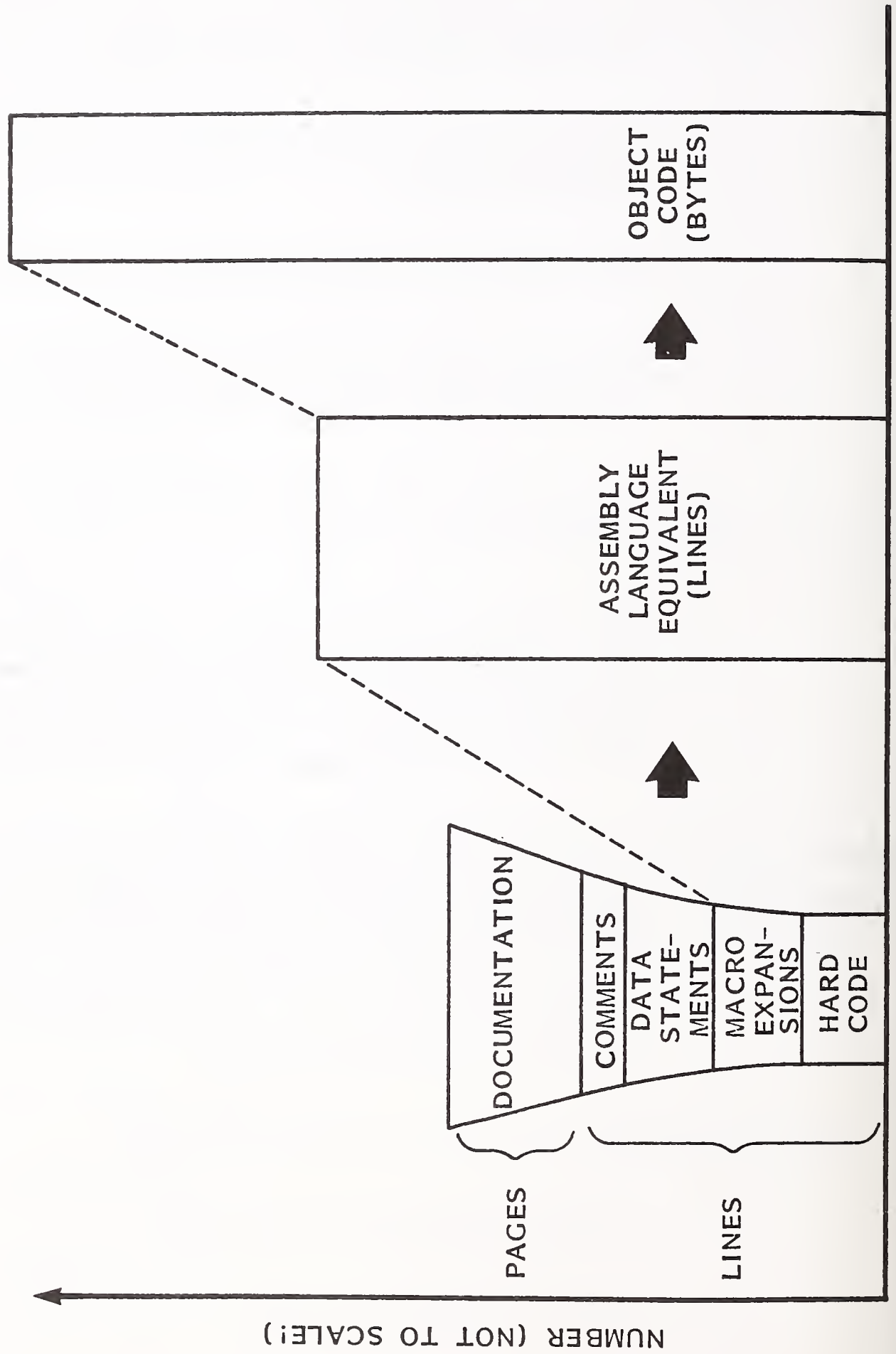
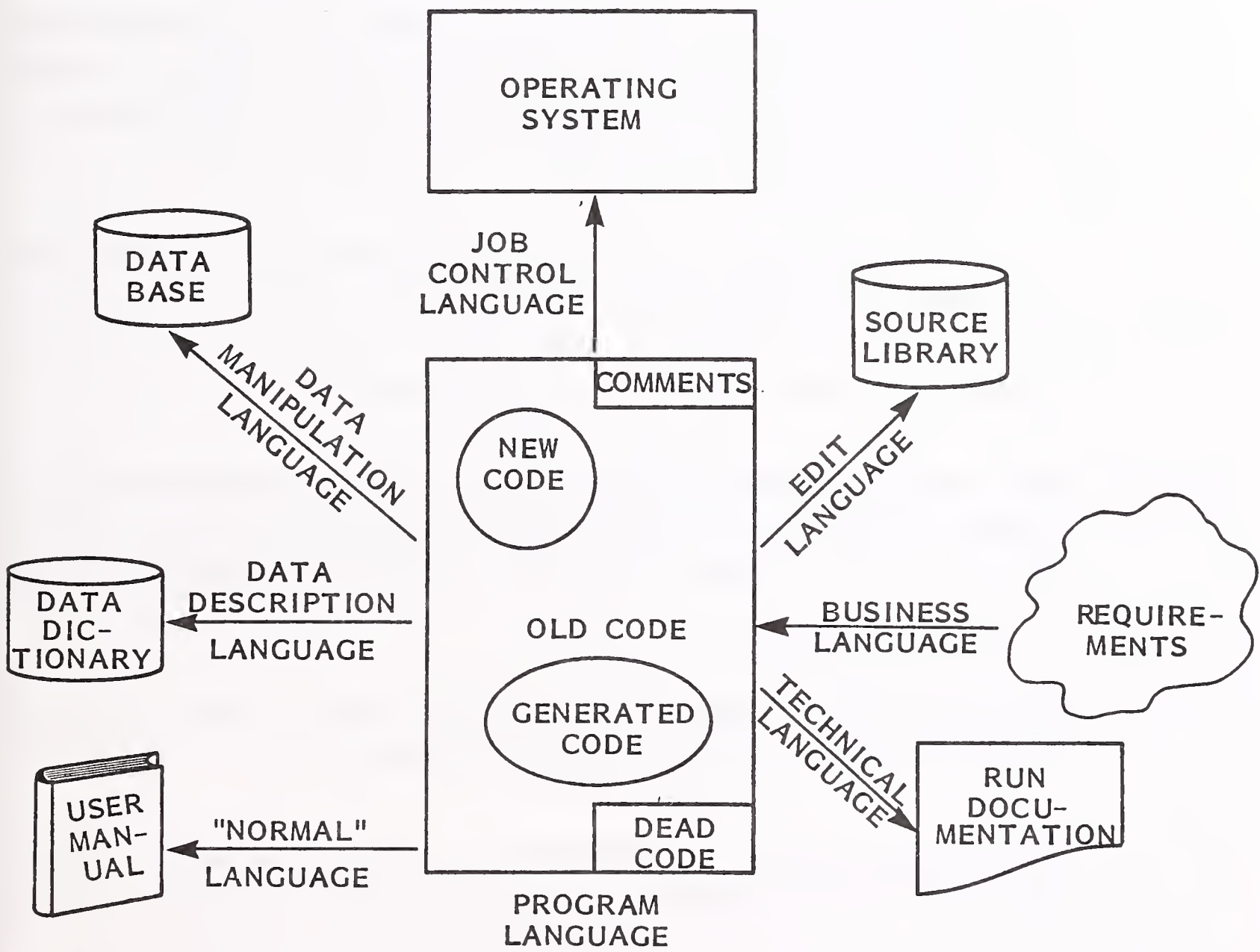


EXHIBIT VII-2

THE MEASUREMENT DILEMMA: MAINTENANCE



WHAT SHOULD BE COUNTED?

- Contrary arguments are not so much against measurement per se, but are warnings not to let the measurement process get out of control:
 - Measurement can become an end in itself, with all attention paid to the symptom and none to the disease. This may lead those practitioners involved to manipulate the measuring system (e.g., produce code in a way that will maximize line counts).
 - Measurement could stifle innovation, since a new technique might appear to be worse than the one it replaced (e.g., the use of program module libraries could produce the apparent effect of a reduction in lines produced per person month).
 - Inadequately chosen (or defective) measurements could produce a false sense of well-being.

2. DESIRABLE CHARACTERISTICS OF MEASUREMENTS

- Assuming that the benefits of measurement outweigh the disadvantages (an assumption each organization will have to evaluate for itself), the following characteristics are proposed as desirable for software implementation productivity measurements.
- Simplicity: A simple measure is better than a complex one since there is less likelihood of errors in both data collection and interpretation.
- Intuitive acceptability: If a measurement is intuitively acceptable, it stands a far greater chance of being used, accepted and properly applied than one that is technically imposing or one that seems to be contrary to common sense.
- Objectivity: This is a truism, but many measurements have built-in subtle assumptions that skew them in a particular direction (e.g., counting pages produced penalizes compact writing).

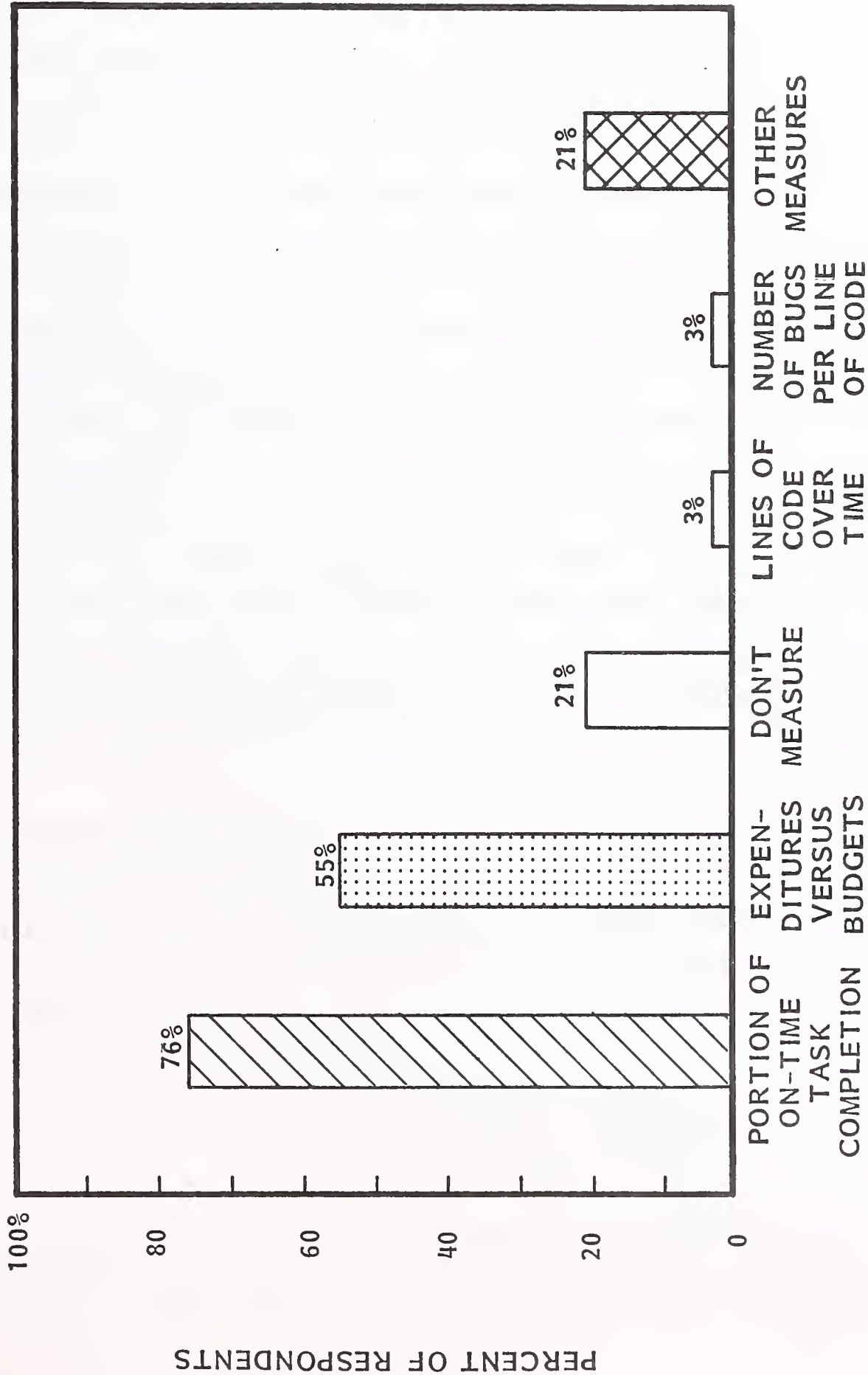
- Unobtrusiveness: A measuring process should not affect the objects being measured (e.g., certain SMF measuring programs add appreciably to the CPU workload).
- Transportability: A measuring process should be transportable from one environment to another (e.g., some measuring tools are dependent on a certain operating system).
- Low cost to produce: Since measurements are supposed to lead to efficiency, they should not add significant amounts to an organization's overhead. Certain personal time reporting systems, by their very comprehensiveness and complexity, add significant amounts to a firm's unproductive overhead (when a sampling system could cost much less and probably give better results).
 - An automated system can often reduce costs unless it encourages the (expensive) collection of more data than can be usefully analyzed and applied.
- Accuracy: A measurement should be accurate; that is, it should correctly measure the underlying event. It will measure absolute quantities, rather than relative amounts.
- Repeatability: A measurement should be reproducible; that is, similar inputs should produce similar outputs in different locations and over different periods of time. This characteristic is also known as "reliability."
- Comprehensiveness: The measurement (or set of measurements) should describe a whole entity, not just one part of it. There is little point to exquisite measurements that address only 5% or 10% of the software development and implementation process.
- Sensitivity: When external changes produce observable differences in a process or product, the measurement should reflect the degree and type of change. Conversely, the measurement should not fluctuate wildly for arbitrarily small values of change. This characteristic is also known as "validity."

- Usefulness: The measurement should deal with topics of management interest that are addressable, not topics where little can be done (e.g., a count of errors occurring within some time period is not as useful as a classification of those errors by type, severity and cause).
- Ability to catch trends: The measurement should reflect underlying trends quickly enough to make adjustments within a relatively short time.
- Ability to facilitate estimates: The measurement should be produced in a form that will enable better estimates to be made (e.g., measurements of lines of code per person month should include adjustments for experience, degree of difficulty, etc.).
- Relation to value: The measurement should relate in some way to the content or value of a function to the organization at large. For example, the value of an accounts receivable system is in the number of days it can shorten the average payment lag, not how many lines of code it contains or even how many transactions it can handle. Thus the cost/benefit ratio of this system, in a productivity framework, is its cost to develop and operate, divided by the number of dollar-days it saves the corporation, figured at some appropriate cost.

3. PRODUCTIVITY MEASUREMENTS CURRENTLY USED

- For whatever reason, most EDP organizations today perform only rudimentary kinds of productivity measurement. When queried about the specific types of measurements they employed, about three-fourths of the responding EDP managers reported that the number of tasks completed on time was their primary productivity measurement, as shown in Exhibit VII-3.
- Expenditures versus budgets are also tracked by about half of the EDP managers, but beyond that there is little uniformity among the measurements employed.

PRIMARY MEASURES OF SOFTWARE DEVELOPMENT PRODUCTIVITY,
AS REPORTED BY EDP MANAGERS



SOURCE: MAIL QUESTIONNAIRE
NUMBER OF RESPONSES = 258

- Only 3% employ measurements relating to lines of code, with frequent dissatisfaction.
- Fully 20% do not bother to measure anything at all.
- It is likely that a lack of understanding about how to use measurements effectively lies at the root of this widespread apathy.
 - Previous unsatisfactory experiences with measurements have also been reported as inhibiting factors in some organizations.
- There are, however, a number of measurements that are relatively easy to use, give good results within their limitations, and enable a company to develop experience in tracking down the elusive factors that contribute to or detract from increased productivity in software development.
- The measurements that are suggested here do not provide an exhaustive list, nor are they without drawbacks. When used as a group, they do provide a relatively comprehensive set of tools to measure progress, identify trends, and track improvements in the development and maintenance process as well as in the software products themselves.
- A number of popular measurements will be presented, together with their strengths and weaknesses. Eleven of them will be rated according to the criteria defined earlier, and classified according to their applicability.
 - Use of the suggested measurements will enable a DP organization to build up a body of data that will, over time, provide a fruitful basis for refinement and analysis of the effects of any productivity initiatives implemented.
 - As long as the data are collected according to a standardized set of definitions, they may then become useful for developing the compara-

tive statistics of cost and time to produce software that every organization desires.

- It should be noted that a few companies, notably IBM and some organizations in the defense and aerospace industries, have been employing measurements for a long time.
 - Examination of these measurements provides little in the way of applicability to the typical commercial environment.
 - Eventually, as software in the commercial environment begins to approach the complexity of software in a computer operating system or a weapons control or space navigation system, some of the knowledge gained by these organizations may become a propos.
 - In the meantime, the learning curve is such that a relatively simplistic set of measurements will suffice to provide more information than most business organizations can comfortably handle right now.
- The Number Of Modules Completed is a basic measurement of progress that can be easily communicated between programmer/analyst, manager and user.
 - Although it frequently happens that the number of modules comprising a system increases between requirements definition and implementation, the curve representing number of modules completed at any point in time, compared to the total number of modules planned, is a tangible indicator of progress.
 - When the two curves begin to converge, it is a sign that all requirements have finally been defined and the end is in sight.
 - For the purpose of this measurement, all modules are given equal weight, without regard to complexity or function. Other measurements can better handle these questions.

- The primary benefit of this measurement is that a completed module is a more tangible entity than a completed task. The definition of a completed module is one that is catalogued in the appropriate library for the state of development that the project is in.
- Percent Of Tasks Completed On Time is a classic measurement that comes closer to the concept of productivity, but actually measures the skill of estimating as much as it does the diligence of performance.
 - Nevertheless, when individual tasks do not exceed one to two weeks in duration, with a defined deliverable to signify the completion of each task, this measure quickly becomes a good indicator of managerial performance.
 - Surveyed organizations that stayed close to these limits in defining tasks report higher probabilities of completing the entire project on time, as well as higher-quality products (because the tasks and their underlying specifications are more definitive).
- Expenditure Versus Budget is a classic measuring tool that is not very useful in a software development environment, if tracked only at the aggregate level.
 - To work, it is dependent on the budget accurately defining the amount of work to be done (or, better, output to be produced) so that actual expenses can be accurately matched against targets. If this is accomplished, then this measurement can be used to evaluate tradeoffs between alternative strategies of development, for example.
 - Most organizations are satisfied to stop short of this level of analysis.
- Lines Of Code Per Unit Of Time and its sibling, cost per 1,000 lines, are the most frequently discussed measurements. The extent of use of these measures is less clear, and the effectiveness of this measurement in real life (as opposed to theory) is even less clear.

- There are definite advantages to counting lines of code.
 - . It relates to other (quality) statistics.
 - . It is a simple, intuitive measure.
 - . It displays the variability attributable to different circumstances of production.
 - . Historical statistics relating to lines of code are usually available.
 - . The measurement of lines of code can be automated.
- On the other hand, there are problems with counting lines of code.
 - . The definition varies by a factor of 2:1 or more.
 - . Performance varies by a factor of 9:1 or more.
 - . Highest-level languages may not use lines.
 - . High-productivity techniques are inherently penalized.
 - . Lines of code measurements focus on new development.
 - . The final software product size does not correlate with counts taken at intermediate steps.
 - . The measure does not account for overhead or other noncoding activities.
- As an example of the complexities of counting, the decision table shown in Exhibit VII-4 outlines the convention that this study recommends to

EXHIBIT VII-4

LINES OF CODE - WHAT TO COUNT

	<u>YES</u>	<u>FIRST TIME ONLY</u>	<u>NO</u>
● NEW EXECUTABLE CODE (1)			✓
● RE-USED CODE			✓
● DATA DECLARATIONS		✓	
● MACRO EXPANSIONS		(2)	
● COMMENTS			✓
● COMPILER DIRECTIVES			✓
● JOB CONTROL STATEMENTS		(3)	
● PROGRAM CHANGES (4)			

(1) 1 LINE = EVERYTHING CONTAINED BETWEEN STATEMENT DELIMITERS

(2) MACRO CALL = 1 LINE EACH TIME USED

(3) AFTER FIRST TIME, COUNTED SAME AS PROGRAM CHANGES

(4) CHANGE TO SINGLE LINE = 1; GROUP OF STATEMENTS INSERTED BETWEEN EXISTING STATEMENTS = 1; CHANGE PLUS INSERTION = 1. DELETIONS, DEBUGGING STATEMENTS, COMMENTS, BLANK LINES = 0.

define a line of code. Unfortunately, reasonable people can disagree on what to count and how to count it, so comparisons between different organizations should be made with extreme caution.

- Performance as expressed in lines of code can also vary greatly.
 - Individuals can vary by as much as 25 to 1.
 - The type of application can have a significant impact as well, with reported variations of over 10 to 1, as shown in Exhibit VII-5.
- Although lines of code produced per unit of time, and cost per 1,000 lines of code, appear to be simple reciprocals, the latter is in fact the superior and preferable form.
 - It accounts more fully for such things as machine time, overhead, cost of debugging, cost of documentation and so forth.
 - Because it is a normalized measure, it may be compared directly to equivalent costs for other systems or other circumstances.
 - It is nevertheless an average, and suffers all of the drawbacks of averages used as measurements.
- Ratio Of Non-Project Time To Total Time is a popular measurement but it is dependent on accurate tracking of personal time, usually by the individuals involved. Variations of a few percentage points are ordinarily studied closely by management; unfortunately, the normal variation is probably greater than this.
 - If the word goes out, employees can produce almost any number that management wants.

EXHIBIT VII-5

"NORMAL" RATE AND VARIATIONS OF L.O.C.

TYPICAL: 1 LINE OF CODE PER HOUR PER PERSON

- EXECUTABLE, DEBUGGED SOURCE
- INCLUDING BUT NOT COUNTING DOCUMENTATION
- SPREAD ACROSS ALL STAFF
- NOT INCLUDING MAINTENANCE

SMALL SYSTEMS: 6-8 OR MORE LINES OF CODE PER HOUR
(LT 2,000 L.O.C.) PER PERSON

LARGE SYSTEMS: $\frac{1}{2}$ OR LESS LINES OF CODE PER HOUR
(GT 500,000 L.O.C.) PER PERSON

- A more typical problem is that employees do not keep good track of their time. If they are forced to do so, the cost of collecting this information becomes unacceptably high.
 - . On the positive side, it is often a useful exercise for both the manager and the employee to discover where the time really goes.
- Turnover Rate can be a useful objective measure of an organization's personnel effectiveness. However, if not coupled with some other tool (such as a structured exit interview), it is likely to raise more questions than it answers.
- Measurements Of Structured Tools: The following measures correlate well with structured tools and approaches, in the sense that they show improved performance from the use of these techniques:
 - Number of job steps is lower with structured techniques.
 - . Fewer unique compilations.
 - . Fewer program test executions.
 - Smaller number of program changes.
 - Smaller program size.
 - . Fewer routines.
 - . Fewer lines of source code.
 - . Fewer control flow decisions.
 - Cyclomatic complexity is lower (to be discussed in the next section).

4. PROMISING MEASUREMENT APPROACHES

- Cost Per Function is a very useful concept in that it tries to measure value (or outputs) rather than costs (or inputs, such as lines of code).
 - One definition of function (stated by A. Albrecht of IBM):
 - Weighted sum of inputs (X4), inquiries (X4), outputs (X5), and master files (X10).
 - Adjusted plus/minus 25% for complexity.
 - A second definition:
 - The number of boxes on the most detailed HIPO chart (approximately equivalent to one page of COBOL code, if using a structured methodology to define a logical function).
 - Advantages to measuring cost per function are that the use of functions as a unit:
 - Facilitates estimating.
 - Remains relatively stable throughout the life cycle.
 - Is large enough to tolerate minor variances without skewing results.
 - Is small enough to be comparable across systems, languages, etc.
 - Relates more closely to value or content of what the system does.
 - Problems with measuring cost per function include:

- . The measurement is not widely used.
 - . It does not relate to other published statistics.
 - . The definition of function is subject to dispute.
- Measuring Defects Per Thousand Lines of Code tracks numbers of defects in the software, but often only in the aggregate. To be truly useful, defects should be classified:
 - By cause.
 - . Design.
 - . Interface.
 - . Data definition.
 - . Logic.
 - . Data handling.
 - . Computation.
 - . Other.
 - By source.
 - . Requirements.
 - . Design.
 - . Coding.

- . Test.
 - . Maintenance.
 - . Corrective maintenance.
- By severity.
 - . Critical.
 - . Major.
 - . Minor.
- These data can then be analyzed to provide a specific plan for improving the quality of software, rather than depending on generalized techniques.
- The advantage of defect removal measurement is that it:
 - Directs attention to quality as an objective.
 - Facilitates analysis of error causes.
- Measuring Cost Per Defect Removed has the advantage of being directly translatable into something that everyone understands; namely, dollars. It also highlights the nonproductive nature of repairing, rather than preventing, defects.
 - There are, however, problems with measuring cost per defect.
 - . High-quality systems are penalized because there will be fewer defects to which fixed defect removal costs can be allocated.

- Defect removal costs are front-loaded, rather than spread over the life of the system.
 - A long time is required to develop baseline data.
 - Baseline changes occur during the maintenance phase.
- Cyclomatic Complexity is a new, largely theoretical measure of a program's complexity. The measure comes from a widely used measure in graph theory, where it equals the number of edges in a directed graph, minus the number of nodes, plus the number of connected components.
 - In a program, it equals the number of predicates (decisions) plus the number of routines.
 - Either is equivalent to the total number of basic control paths through a program.
- This measure relates well to predicted effort to develop and maintain a program, and has been used to determine that a given design was not likely to succeed because it was too complex.
 - The measure is more useful after coding is complete than before coding has begun, but it nevertheless furnishes useful information about the quality of the product.

5. COMPARATIVE ANALYSIS OF MEASUREMENT APPROACHES

- The beginning of this section described measurement attributes that are applicable to different situations. These are:
 - Process.
 - Product.

- Cost.
 - Rate.
 - Development.
 - Maintenance.
 - Operation.
 - Individuals.
 - Systems.
- The desirable characteristics of measurements were described as:
 - Simple.
 - Intuitive.
 - Objective.
 - Unobtrusive.
 - Transportable.
 - Inexpensive.
 - Automatable.
 - Accurate.
 - Repeatable.

- Comprehensive.
 - Sensitive.
 - Able to catch trends.
 - Able to aid estimates.
 - Useful.
 - Related to value.
- Finally, in the previous two sections, current and promising measurement approaches were described and analyzed. They included:
 - Number of modules completed.
 - Percent of tasks completed on time.
 - Expenditures versus budget.
 - Lines of code per hour, month, etc.
 - Cost per 1,000 lines of code.
 - Cost per function.
 - Bugs per 1,000 lines of code.
 - Percent of bugs removed.
 - User satisfaction.
 - Non-project time.

- Total time.
- Turnover rate.
- Each measurement can be ranked high, medium or low in desirability and applicability. Exhibit VII-6 arrays each measurement and assigns a rating for each characteristic and attribute.
- A particular data processing environment may wish to emphasize measurements that are particularly strong in certain characteristics, but not so critical in others.
- EDP measurements are an evolving subject where there can legitimately be differing opinions. These ratings are offered as a starting point.
- Another way of looking at this complex of relationships is to assign a numeric value to each rating (H=3, M=2, L=1) in order to derive total scores for desirable characteristics and applicable attributes. These scores are an indication of a measurement's all-around utility. Exhibit VII-7 plots these scores for each measurement, with higher scores placed nearest to the upper right-hand corner.
- The limited scope of lines-of-code-per-time-unit and defect-removal measurements shows up clearly on this chart.
- It should be stressed that no weights were assigned to the various factors. Consequently, the actual utility of a given measurement to a particular firm could be much different from that shown here.

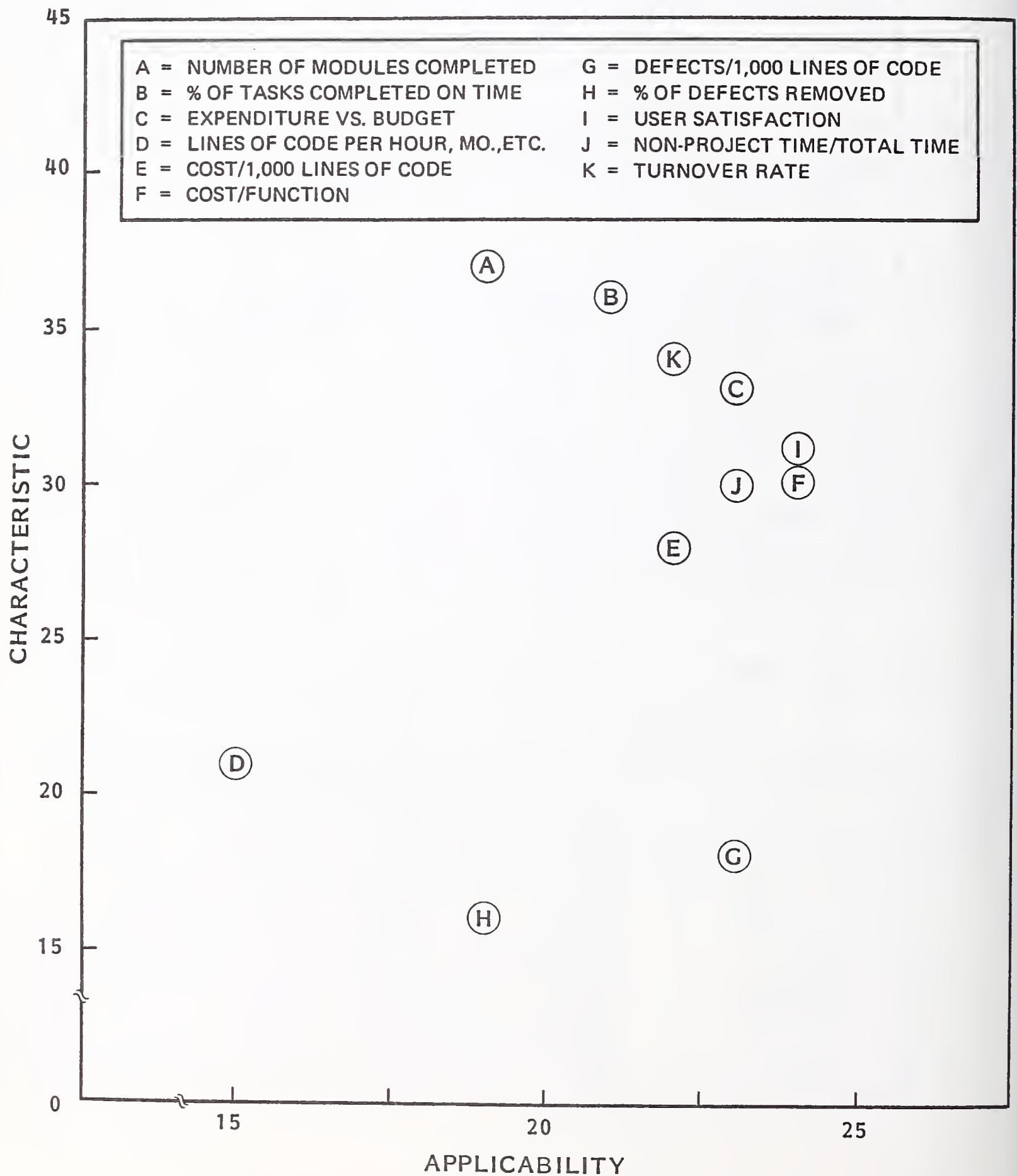
EXHIBIT VII-6

RATING OF SOFTWARE IMPLEMENTATION MEASUREMENTS

	CHARACTERISTIC														APPLICABILITY									
	SIMPLE	INTUITIVE	OBJECTIVE	UNOBTRUSIVE	TRANSPORTABLE	INEXPENSIVE	AUTOMATABLE	ACCURATE	REPEATABLE	COMPREHENSIVE	SENSITIVE	CATCHES TRENDS	AIDS ESTIMATES	USEFUL	RELATED TO VALUE	PROCESS	PRODUCT	COST	RATE	DEVELOPMENT	MAINTENANCE	OPERATION	INDIVIDUAL	SYSTEMS
H = HIGH M = MEDIUM L = LOW - = NOT RATED	(A)	NUMBER OF MODULES COMPLETED	(B)	% TASK COMPLETED ON TIME	(C)	EXPENDITURES VERSUS BUDGET	(D)	LINES OF CODE PER HOUR, MONTH, ETC.	(E)	COST/1,000 L.O.C.	(F)	COST/FUNCTION	(G)	BUGS/1,000 L.O.C.	(H)	% BUGS REMOVED	(I)	USER SATISFACTION	(J)	NON-PROJECT TIME TOTAL TIME	(K)	TURNOVER RATE		
	H	H	H	H	M	H	H	H	H	L	M	M	M	M	M	M	H	L	L	H	M	L	M	H
	H	H	H	M	M	H	M	H	M	L	M	M	M	M	M	M	M	L	L	H	M	L	M	H
	H	H	M	H	L	M	M	L	L	L	M	M	M	M	M	M	M	M	M	H	H	L	M	H
	H	M	L	L	L	M	M	L	L	L	M	M	M	M	M	M	M	L	L	H	H	-	M	H
	M	M	M	L	L	M	M	M	M	M	M	M	M	M	M	M	M	H	H	H	H	M	M	H
	M	M	M	L	L	M	M	M	M	M	M	M	M	M	M	M	M	H	H	H	H	M	M	H
	M	M	M	L	L	M	M	M	M	M	M	M	M	M	M	M	M	H	H	H	H	M	M	H
	M	M	M	L	L	M	M	M	M	M	M	M	M	M	M	M	M	H	H	H	H	M	M	H
	M	M	M	L	L	M	M	M	M	M	M	M	M	M	M	M	M	H	H	H	H	M	M	H

EXHIBIT VII-7

RANKING OF PRODUCTIVITY MEASUREMENTS



VIII RECOMMENDATIONS

VIII RECOMMENDATIONS

A. OVERVIEW

- Many recommendations for improved software productivity have been made, either explicitly or implicitly, throughout earlier sections of this report.
- However, it will be useful to managers if INPUT's recommendations are summarized in one place, with references back to the body of the text for more detail.
- Recommendations are of two kinds.
 - Strategic recommendations, which involve:
 - The management and organization of the entire company as well as the EDP department.
 - The relationship of EDP to individual users and top management.
 - Long-term EDP planning.
 - Tactical recommendations, which involve:

- . Approaches to retaining and maximizing the value of DP personnel.
 - . Approaches to systems development and project management.
 - . Programming and design aids.
- Tactical recommendations can be implemented to a large extent on the sole initiative of EDP management, although external approvals and/or cooperation may be required for expenditures, personnel policy changes and so forth.
- Strategic recommendations usually involve the active cooperation of groups outside the EDP department and often involve changes in the way these other groups conduct their work.
- A conclusion that has become painfully obvious during the course of this study is that there are very few universally applicable productivity improvement recommendations.
 - Organizations differ greatly in their corporate characteristics as well as in their stages of productivity development.
 - What is hypothetically appropriate for a particular organization may turn out to be pragmatically tenable.
- As stated in the introduction to this report, the study's objective was to provide a framework and perspective on all of the major factors that influence software productivity:
 - To categorize their interactions and their relative degrees of impact on productivity.
 - To provide an instrument to diagnose any EDP organization's current stage of productivity development.

- To furnish background material and guidance for assessing the various categories of productivity improvement available as well as their suitability, costs and benefits.
- To direct the EDP manager in establishing a strategic productivity improvement plan.
- Certain principles are preeminent in establishing this plan. They include:
 - The primacy of the user, and the necessity for keeping the interface between user and information as short, direct and immediate as possible.
 - Facilitating access to resources for software development and maintenance staff.
 - Automating development and maintenance tasks (or parts of those tasks) that are repetitious, tedious, complex or otherwise suitable for computer assistance.
 - Asserting the priority of quality, reliability and maintainability as design objectives and evaluation criteria.

B. DEVELOPING A STRATEGIC PLAN

- Because the overwhelmingly large increases in productivity can come only from making the entire corporation more effective through better use of better information, every EDP productivity improvement effort should be implemented with this overall goal in mind.
 - The end user must be intimately involved with all phases of the process that turn raw data into powerful, timely, accurate business information.

- As much as possible, the intermediary translations and interpretations of specifications and requirements should be eliminated, and the user should be placed firmly in control of the information needed to conduct, plan, and evaluate his or her business functions.
- The EDP organization, however, must remain in control of the process of furnishing that information in coordination with the needs and priorities of the entire corporation.
- Strategies for implementing these principles are highly dependent upon an individual organization's characteristics. The first step, then, in developing a software productivity improvement strategy is to conduct an evaluation of the EDP organization's current stage of productivity development.
 - Use the self-analysis matrix described in Chapter IV to establish the predominant productivity stage, based on the characteristics defined in that instrument.
 - Determine specific strengths and weaknesses by applying the detailed audit contained in Appendix E.
 - Analyze the improvement priorities required with the priority matrix given in Exhibit IV-13.
 - Modify these priorities, if necessary, based on unique local conditions and the point scores obtained from the detailed activity audit.
- Using the results obtained above, consult Chapter V to determine the appropriate category of aids to address the needs and priorities determined so far.
- Prepare an evaluation and implementation plan based on the experiences of other organizations with these tools, as described in Chapter VI.

- Incorporate measurements and assessments as appropriate from Chapter VII.
- Implement the plan.
- After a suitable period of time, usually six months to a year later, again apply the self-analysis matrix and the detailed audit to determine what progress has been made, and which areas now need improvement.

C. SPECIFIC STRATEGIC RECOMMENDATIONS

- Quality should be introduced as an explicit management goal as early as possible.
 - This last point should be stressed. There are times in the development of EDP organizations when proper value will not be placed on quality, e.g.:
 - In the "Chaos" and early "Control" stages.
 - When the non-EDP organization does not stress quality.
 - When EDP management is under intense pressure to meet time or cost objectives, and very short-term goals predominate.
 - See Sections III.B and IV.C.
- User involvement should be fostered at all times (III.C).
 - If an organization is in the "Chaos" or "Control" stage (or oscillating between them), then constructive user involvement is difficult.

- However, attempts to involve users at levels appropriate to the EDP stage of productivity development are valid and often productive in any stage. (See Section IV.C and Appendix D.3).
- DP management has the ultimate responsibility for productivity improvement, and should take the initiative in becoming acquainted with, and involved in, corporate-wide policies and issues.
 - This represents a distinct and necessary departure from the widespread present situation where DP management sees the corporate environment as the least important factor in productivity (III.B.6).
- Similarly, EDP management should construct a realistic (i.e., practicable) long-range plan, as shown in Appendix E.2).
 - By making its assumptions explicit, EDP management will often find that it is not in step with the company's own long-range plan, as shown in III.B.
 - A long-range plan, together with ongoing modifications, is an excellent method of maintaining a useful dialogue with users and top management.
 - Planning should be appropriate for the organization's stage of productivity development and its capability for planning, as shown in Appendix E.2.
- An EDP steering committee at an appropriate level (or levels, in the later stages) will assist in ensuring that the EDP department is doing productive work, as shown in Appendix E.2.
 - The level of involvement must match, or slightly lead, the organization's corporate management style.

- It should not be a sham (run by the EDP Director) or a paper tiger (never saying no).
- A company-wide consciousness of productivity is a most useful adjunct to EDP productivity efforts.
 - It is lacking in most firms now (III.B.3).
 - EDP management could play a constructive role in changing this situation.
 - By doing so, the EDP organization would become cognizant of the larger picture; i.e., the total corporation's productivity issues.

D. TACTICAL RECOMMENDATIONS

- Develop good managers and give them leeway to operate. A Management by Objectives program is an excellent means of accomplishing this, as shown in III.E.1 and III.G.
- If not already in place, implement a system development methodology, as shown in V.C.8.
- Establish a workable project control system, as shown in Section VII.A.3.
 - Its importance for managing projects so that they come in on time and budget should be obvious.
 - That it is equally important for analyzing why projects did not meet time/budget estimates, so that the estimation process can be improved, is overlooked by most organizations.

- It should track maintenance time by cause and source of error, as shown in Section VII.D.4.
- Establish parallel career paths for managerial and technical capabilities, as shown in III.E.5.
 - Convey recognition and delegate responsibility to each personnel level.
 - Prepare well-defined goals in conjunction with staff, not top-down.
 - Concentrate on increasing the skills of each person in the EDP organization.
- Develop defensive recruitment approaches, as shown in III.E.3.
 - Focus on internal upgrading and transfers.
 - Disperse tasks to users.
 - Employ graduates of two-year programs.
 - Assist local colleges with curriculum development, lecturers, work/study programs, etc.
 - Consider a trainee-rich organization.
 - Don't be bound by the concept of only a 9-to-5, full-time staff.
 - When hiring, don't sacrifice quality and personal characteristics of the individual (stability, dependability, attention to details, etc.) to experience (CICS, database or whatever). Skills can be taught. Behavior cannot.

- Re-establish the role of business analyst in user department/divisions, as shown in Section III.E.
 - The functions of this individual are to act as a liaison, to facilitate internal prioritization of needs, and to serve as an identifier and provider of user training.
 - It will frequently be easier and more effective to teach systems analysis techniques to a person who understands business, than to teach business to someone who understands systems analysis.
- Use on-line development, as shown in Appendix E.4, accessing an interactive library of reusable modules (V.B.5).
- Use computer-aided design tools wherever appropriate, as shown in Appendix E.5. Consider developing such tools if they are not commercially available or feasible to purchase.
- Use structured programming techniques in high-level languages, as shown in Appendix E.5.
- Use software packages where cost/effective (V.B).
- Use the most qualified staff for critical processes (system design, common code, etc.), reviewing systems for weaknesses.
- Establish a quality assurance group.
 - Where available, use automated testing tools (Appendix E.5).
 - Insist that programmers use the highest-level tools available, consistent with the task performance specifications.

- Test to expose the system's weaknesses as well as to demonstrate its proper operation.
- Don't treat maintenance as a step-child (Appendix E.5).
 - Use the same techniques and standards as in new development.
 - Integrate system and data flow descriptions with program listings.
 - Use automated tools, where appropriate.
 - Keep as much documentation on-line as possible.
 - Have a maintenance plan and control changes.
 - Maintain documentation and error logs.
 - Choose a design technique for new development that minimizes maintenance over the life cycle.
- In later development stages, interate the various aids and tools available (IV.C).
 - While there will be new integrated tools and aids coming on the market in the next few years, in many cases a competent systems programmer can develop the necessary interfaces to make the various tools work together.
 - In some cases, all that is necessary is a "buddy" training program that lets more junior staff see how to use the tools in an effective way; e.g., staging a single compilation/test run that has its results automatically compared to expected results, and the exceptions routed to the programmer's terminal for analysis and debugging.

E. RECOMMENDATIONS FOR SMALL ORGANIZATIONS

- For the smaller DP organization, some of the recommendations may sound hopelessly "blue sky," and in fact may not be appropriate for this size firm.
 - The \$150,000 software package, for example, may seem impossibly out of reach. But the real costs to analyze in cases like these are the costs of not having the package; i.e., of not being able to perform the desired function in an adequate or timely way - of not having this capability available.
- To an even larger extent than in large organizations, timing of productivity improvements in small organizations is critical. Extreme attention must be paid to finding the application, the situation, for which the benefits of the new techniques are so obvious that its implementation is beyond question.
 - The "efficiency" class of improvements assumes greater importance for small organizations. Facilitating access to resources produces immediate, obvious benefits.
 - Any of the "shorthand" tools will likely have a high payoff, and even such mundane approaches as typing classes for programmers can lead to quick improvements.
- The single fact of being a small organization is not an automatic barrier to productivity. Many of the most effective productivity tools now available started out as the brainchildren of a single, desperate or frustrated individual who had no choice but to become more efficient.
 - The distinct advantage of a small organization is that it is usually able to control its own resources without an excessive amount of outside interference or bureaucratic procedures.

APPENDIX A: INTERVIEW PROGRAM

APPENDIX A: INTERVIEW PROGRAM

PRODUCTIVITY IMPROVEMENT IN SYSTEMS AND SOFTWARE DEVELOPMENT

RESEARCH PROGRAM

CATEGORY	VENDORS	USERS					OTHER ORGAN- ZATIONS
		COMPANIES	MANAGERS	PROGRAM/ PROJECT LEADERS	ANALYST/ PRO- GRAMMERS	TOTAL IN- DIVIDUALS	
ON-SITE	13	40	55	124	45	224	10
PHONE	10	-	-	-	-	-	22
MAIL QUESTIONNAIRE	-	400	400	-	-	400	-
EDP USER PANEL	-	900	900	-	-	900	-
TOTAL	23	1,340	1,355	124	45	1,524	32

APPENDIX B: BIBLIOGRAPHY

APPENDIX B: BIBLIOGRAPHY

- This is by no means a complete - or even comprehensive - bibliography. However, it does give an annotated sampling of papers and books on a wide variety of topics related to EDP productivity, ranging from "classics" that first broached the idea of structured programming, to the latest results in such experimental fields as automated program validation.
- The Data & Analysis Center for Software (DACS) (operated by IIT Research Institute for the Rome Air Development Center, RADC/ISISI, Griffiss AFB, NY 13441 ((315) 336-0937), has collected an extensive library of statistics and documents related to software reliability and productivity. DACS publishes a newsletter and provides retrieval services and some other products related to their data base free of charge. They also seek information to be added to their data base from any organization that is in a position to contribute to their efforts. For further information, contact Mr. Gary F. Caron, Program Manager.
- Two publications by the IEEE Computer Society contain reprints of many of these papers. They are:
 - Tutorial on Software Design Techniques. Edited by Peter Freeman and Anthony I. Wasserman. Third edition, 1980, 455 pages. IEEE Catalog No. EHO 161-0.

- Tutorial: Automated Tools for Software Engineering. Edited by Edward Miller. Papers initially presented at Comp-Sac 1979. 262 pages. IEEE Catalog No. EHO 150-3. Many of the papers contain extensive bibliographies.
- Both may be obtained from the IEEE Computer Society, at either 5855 Naples Plaza, Suite 301, Long Beach, CA 90803, or at 445 Hoes Lane, Piscataway, NJ 08865.
- Sources for informational materials on some of the commercially available tools and aids are included in the Glossary, Appendix C of this report.
- A selected list of articles, books and papers follow:

Agerwala, Tilak. "Putting Petri Nets to Work," Computer (IEEE Comp. Soc.). December 1979, pp. 85-93.

Baker, F. Terry. "Structured Programming in a Production Programming Environment," IEEE Trans. on Soft. Engrg. June 1975.
(Comment: IBM/FSD use of IPT.)

Baker, F. Terry. "Chief Programmer Team Management of Production Programming," IBM Systems Journal. Vol. 11, No. 1, 1972, pp. 56-73.

Bell, Thomas E.; Bixler, David C.; and Dyer, Margaret E. "An Extensible Approach to Computer-Aided Software Requirements Engineering," IEEE Trans. on Soft. Engrg. January 1977.
(Comment: RSL/REVS.)

Boehm, Barry W. "Software and its Impact: A Quantitative Assessment," Datamation. May 1973.
(Comment: Classic.)

Brooks, Frederick P. Jr. "The Mythical Man Month," Datamation. December 1974, pp. 45-52.

(Comment: Classic; a more complete version published by Addison-Wesley, Reading, MA, 1975. The December 1974 Datamation also has several other articles of interest.)

Budd, Timothy A., et al. "The Design of a Prototype Mutation System for Program Testing," Proc. Nat'l Computer Conf., Vol. 47, 1978, AFIPS Press.

Caine, Stephen H. and Gordon, E. Kent. "PDL - A Tool for Software Design," Proc. National Comp. Conf., Vol. 44, 1975; AFIPS Press.

Dahl, O.J.; Dijkstra, E.W.; and Hoare, C.A.R. Structured Programming. Academic Press, London, 1972.

(Comment: Classic book.)

Davis, David P. and Tweedy, Kevin F. "Chevron's Integrated and Automated Approach to Applications Development," Proc. Conf. on Application Development Systems. Santa Clara, March 1980. Pub. in Data Base, Winter-Spring 1980.

DeMarco, Tom. Structured Analysis. Yourdon Press, New York, 1968.

(Comment: Also, Structured Analysis and System Specifications, Prentice Hall, Englewood Cliffs, NJ, 1979.)

Dolotta, T.A. and Haight, R.C. PWB/UNIX - Overview and Synopsis of Facilities. Bell Telephone Labs, Inc., June 1977.

Dolotta, T.A. and Mashey, J.R. "An Introduction to the Programmer's Workbench," Proc. 2nd Int'l Conf. on Soft. Engrg. October 1976, Pub. by IEEE.

Fagan, M. "Design and Code Inspections to Reduce Errors in Program Development," IBM System Journal. Vol. 15, No. 3, 1975.

Gane, Chris and Sarson, Trish. Structured System Analysis: Tools and Techniques. Prentice Hall, Englewood Cliffs, NJ, 1979.

Howede, William E. "DISSECT - A Symbolic Evaluation and Program Testing System," IEEE Trans. on Soft. Engrg. January 1978.
(Comment: Symbolic testing.)

IBM. Improved Programming Technology - An Overview. IBM Corp., Document No. GC20-1850.

IBM. HIPO - A Design Aid and Documentation Technique. IBM Corp., Document No. GC20-1851.

Jackson, Michael A. "Constructive Methods of Program Design," Proc. First Conf. of the European Cooperation in Informatics. Springer-Verlag, 1976, pp. 236-262.

Jackson, Michael A. Principles of Program Design. Academic Press, 1975.

Jones, T. Capers. "Measuring Programming Quality and Productivity," IBM System Journal. Vol. 17, No. 1, 1978.

Lanergan, Robert G. and Poynton, Brian A. "Reusable Code - The Application Development Technique of the Future," Proc. Application Development Symposium, Monterey, CA. October 1979. Guide/Share/IBM.
(Comment: This publication has 24 other papers of interest.)

Lientz, Bennet P. and Swanson, E. Burton. "Impact of Development Productivity Aids on Applications Systems Maintenance," Data Base. Winter-Spring 1980, pp. 114-120.

Love, Tom and Fitzsimmons, Ann. "A Review and Evaluation of Software Science," ACM Computing Surveys. Vol. 10, No. 1, March 1978, pp. 4-18.
(Comment: Halstead's Software Metrics; 41 item bibliography.)

Miller, G.A. "The Magical Number Seven Plus or Minus Two," Psychology Review, 1956, pp. 81-97.

(Comment: Classic.)

Mills, Harlan D. "On the Development of Large Reliable Programs," Proc. IEEE Symposium on Computer Soft. Reliability. 1973, Pub. by IEEE.

Noe, J.D. "A Petri Net Model of the CDC 6400," Proc. ACM SIGOPS Workshop on System Performance Evaluation. 1973, pp. 362-378.

Orr, Kenneth T. Structured Systems Development. Yourdon Press, New York 1977.

Osterweil, Leon J. and Fosdick, Lloyd D. "DAVE - A Validation Error Detection and Documentation System for Fortran Program," Software: Practice & Experience. October - December 1976.

(Comment: Static analysis.)

Peterson, J. "Petri Nets," ACM Computing Surveys. Vol. 9, No. 3, September 1977, pp. 223-252.

Ramamoorthy, C.V. and Ho, Siu-Bun F. "Testing Large Software with Automated Software Evaluation Systems," IEEE Trans. on Soft. Engrg. March 1975.

Reifer, Donald J. and Trattner, Stephen. "A Glossary of Software Tools and Techniques," Computer (IEEE Comp. Society). July 1977.

(Comment: 61-item bibliography.)

Rochkind, Marc J. "The Source Code Control System," IEEE Trans. on Soft. Engrg. December 1975.

(Comment: PWB/UNIX.)

Ross, Douglas T. "Structured Analysis (SA): A Language for Communicating Ideas," IEEE Trans. on Soft. Engrg. January 1977.

(Comment: SADT.)

Ross, Douglas T. & Schoman, Kenneth E. "Structured Analysis for Requirements Definition," IEEE Trans. on Soft. Engrg., January 1977.

(SADT; Ross is regarded by some as the true originator of structured analysis which Yourdon, Gane, etc., followed.)

Sackman, Harold. Man-Computer Problem Solving. Auerbach Publishers, 1970.

(Comment: Book - a rare report on batch versus on-line experiments.)

Stay, J.F. "HIPO and Integrated Program Design," IBM Systems Journal, Vol. 15, No. 2, 1976.

Stevens, W.P.; Myers, Glen J.; and Constantine, Larry L. "Structured Design," IBM Systems Journal. Vol. 13, No. 2, 1974.

Teichroew, Daniel and Hershey, Ernest A. III. "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems," IEEE Trans. on Soft. Engrg. January 1977.

Walston, C.E. and Felix, C.P. "A Method of Programming Measurement and Estimation," IBM Systems Journal. Vol. 16, No. 1, 1977.

Weinberg, Gerald M. The Psychology of Computer Programming. Van Nostrand Reinhold Co., 1972.

(Comment: Book. Weinberg is the originator of "egoless programming.")

Weinberg, Victor. Structured Analysis. Yourdon Press, New York.

Wirth, Niklaus. "Program Development by Stepwise Refinement," Comm. of the ACM. April 1971.

(Comment: Classic.)

Yoder, Cornelia M. and Schrag, Marylin L. "Nassi - Schneiderman Charts: An Alternative to Flowcharts for Design," Proc. ACM SIGSOFT/SIGMETRICS Software and Assurance Workshop. November 1978. Pub. by ACM.

APPENDIX C: GLOSSARY

APPENDIX C: GLOSSARY

This Glossary contains definitions and descriptions of selected tools, aids, techniques and methodologies relating to applications development, testing and operation.

- **ADF - APPLICATIONS DEVELOPMENT FACILITY** This is an optional IBM IUP (Installed User Program) that works in conjunction with IMS and provides a menu-driven programming environment. Dramatic improvements in programmers' productivity (better than 10 to 1) have been reported for ADF, at the cost of about 50% more CPU time than equivalent IMS applications coded in COBOL. See Menu-Driven Programming.
- **ADF - AUTOMATED DESIGN FACILITY** An optional component of PRIDE-Logik.
- **ASDM - AUTOMATED SYSTEM DESIGN METHODOLOGY** An alternative name to the PRIDE-Logik system.
- **BIAIT - BUSINESS INFORMATION ANALYSIS AND INTEGRATION TECHNOLOGY** BIAIT, developed by Donald C. Burnstine of BIAIT International, Petersburg, NY, is a formal discipline that provides structure and reproducibility in the problem analysis phase of business-oriented data processing work. The key concept underlying BIAIT is that the information-handling processes involved in many types of businesses can be described in terms that are descriptive of business information handling rather than EDP-related functions. A model of the business is expressed in a BICMX (Business Information Control Matrix), which arrays business functions against the

"inventories" that each function keeps track of. Every "inventory" can be broken down into steps of its generic information life cycle. Within the matrix, users, suppliers, authority and responsibility are interrelated by information interfaces. The system has had some use within and outside IBM.

- **BOTTOM-UP IMPLEMENTATION** This term is used both to describe the chaotic, undisciplined situation prevailing prior to "structured" design processes; and as the name of a legitimate design and/or implementation technique. In a bottom-up implementation, for instance, a system design, which may have been performed using top-down techniques, is analyzed to determine the set of most elementary functions that are specified or required; these are implemented and tested first. Then successively higher levels of design are implemented, using the modules previously coded and proved. Bottom-up is the most natural way to implement systems where reusable code libraries play a major role.
- **CARA** is a commercial, manual system design methodology contained in two volumes, a general reference card and preprinted forms. The system defines what tasks must be done and when, the deliverables at the conclusion of each life cycle phase, and the documentation standards. The system is priced at \$22,500 and is being marketed by CARA Corp., 1010 Jorie Blvd., Suite 124, Oak Brook, IL 60521, (312) 654-2405.
- **CHIEF PROGRAMMER TEAM** An organizational structure for the development of software systems, developed by the Federal Systems Division of IBM in 1969-1971 and later released as part of the Improved Programming Technologies package. In this scheme, software is developed by small teams, each consisting of a minimum of three and a maximum of five persons. The team is headed by the Chief Programmer, who does the design, codes the highest level of implementation and supervises the work of additional programmers who code "stubs" (subroutines or lower levels). The Backup Programmer aids the Chief Programmer and is ready to assume the role of Chief Programmer, should it become necessary. The Programming Secretary performs the clerical work according to a highly disciplined routine that makes

all source listings, computer run results and data "publicly available" in a Programming Production Library (PPL) (also called DSL - Development Support Library) and in archive records. The CPT system relies on structured programming and top-down design concepts, and is claimed to have resulted in doubling programmers' productivity and in enhanced reliability and maintainability of the resulting code. IBM reported that a five-person CPT is expected to produce up to 25,000 lines of source code in 12-18 months.

- **CICS - CUSTOMER INFORMATION CONTROL SYSTEM** An IBM program product, CICS is generally classified as a telecommunications monitor. Like TSO, to the operating system it appears to be just another applications program. CICS, however, contains a supervisor that manages the resources allocated to it and distributes them among multiple, terminal-driven, transaction processing tasks. These tasks are user-coded; they need not be concerned with the telecommunications aspects of controlling the terminals - CICS performs this function. Numerous competing products are available, including SHADOW from Altergo (now INSAC SOFTWARE, INC.); ENVIRON I from Cincom; GBASWIFT from GBA International; BETACOMM, INTERCOMM and MINICOMM; DATACOM/DC from ADR; TP2000 from Intel/MRI; COMPLETE from Software ag; TASK/MASTER from Turnkey Systems/NCSS; WESTI from Westinghouse; and IBM's own IMS/DC.
- **CMS - CONVERSATIONAL MONITOR SYSTEM** CMS is a "native" component of the VM/370 operating system for IBM mainframes. It is oriented to the support of a single terminal, engaged in program development and debugging activities. By running multiple "CMS machines," a system under VM/370 can provide the same type of interactive, multi-user, program development environment that is the object of TSO under MVS.
- **CODASYL** Conference on Data System Languages is a U.S. government-sponsored, voluntary organization formed in 1959 to further the development of data base languages and systems.

- **CONSTANTINE/DEMARCO/YOURDON METHOD** This structured analysis methodology is based on ideas first proposed by Larry A. Constantine, W.P. Stevens and Glenford J. Myers (all of IBM), and later refined and developed cooperatively by Tom DeMarco and Edward Yourdon. A "Data Flow Diagram" or "Bubble Chart" is used to discover and describe the changes that each input must undergo in order to produce a desired output. A "Structure Chart" is then developed to show the hierarchy of the changes or actions. The method can therefore be classified as data-driven. The Structure Chart is translated into pseudo-code, which can then be turned into executable code. Key concepts in the method include cohesion, coupling, and span of control and effect; they are used to judge the modularity of the resulting design, its correctness and completeness.
- **CORRECTNESS PROOFS** Correctness proofs are attempts to demonstrate the "correctness" of programs by rigorous mathematical methods. "Correctness" usually means the ability of a program to carry out a specified task and only that task; i.e., without errors. The most common approach to this problem has been to define the program specifications and the program steps in terms of function theory, a mathematical discipline. Structured programs are especially well-suited to being stated in such terms. The method has so far proven impractical for large programs, because the effort involved in creating the proof is, as a rule, far greater than the effort required to debug the program by empirical techniques.
- **CPT** See Chief Programmer Team.
- **DBMS - DATA BASE MANAGEMENT SYSTEM** A software system intended to centralize the creation, control and maintenance of data files, so that multiple application programs can access the data without having to create duplicate private file systems. DBMS generally involves a Data Definition Language - DDL - to create the data base, which is typically under the control of one person or department, called the data base administrator. The accessing of the data base is done through a Data Manipulation Language (DML) which can be embedded in a programming language like COBOL, or can

be a standalone language, as in many data base query/retrieval languages. A great deal of standardization work in the area of data base management has been done by the CODASYL Data Base Task Group. A well-designed data base with an associated on-line query/retrieval system generally results in significant reductions in the number of special coding projects that a DP department is required to do to satisfy user needs for specialized reports. The term DBMS is used by DEC as the name of a specific data base system for the PDP-10 and its successor 36-bit systems.

- **DeMARCO, TOM** Author of a structured analysis methodology. See Constantine/DeMarco/Yourdon.
- **DMS - DEVELOPMENT MANAGEMENT SYSTEM** DMS is a menu-driven program development system from IBM that permits nonprogrammers to create useful systems by supplying "fill-in-the-blank" specifications either interactively, via 3270 terminals, or off-line, via specially printed forms. The system relies on CICS for terminal monitoring and control, and is available for IBM mainframes under DOS or OS environments. A version of DMS is also available on the IBM 8100 and 3790 systems. See Menu-Driven Programming.
- **DSL - DEVELOPMENT SUPPORT LIBRARY** DSL is a formalized computerized archiving system developed by IBM's Federal Systems Division, which takes the view that all computer run results, source listings, test data, etc., are "public records" that should be highly visible to both project personnel and management. This is in contrast with the more typical practice where individual programmers keep private records and the results of "bad" runs - which may contain useful data for the project - are often discarded. DSL was made formally a part of the Chief Programmer Team concept (see entry) under the name PPL - Programming Production Library.
- **FOCUS** is an integrated data base management system and query language. It requires TSO or CMS for terminal control. It maintains its own data files, although IMS files can be operated on as an option for the TSO version. In addition to the data management facility and report generator, FOCUS also

incorporates a transaction processor. The query language is close to natural English. Optional features include a host language interface (to allow access to FOCUS files from COBOL, Fortran, PL/I or BAL); a graphics subsystem; statistical analysis and financial modeling packages; and the IMS interface. FOCUS is marketed by Information Builders Inc., 254 West 31st St., New York, NY 10001, (212) 736-4433. Prices begin at \$46,000 for the basic package, including data management, report generator and transaction processor.

- **FUNCTIONAL DECOMPOSITION** When used in conjunction with design or programming methodologies, this term implies a top-down design technique (see entry). The term actually originated in the theory of functions - a mathematical discipline - where it has a more limited and precise meaning.
- **GANE & SARSON** A design methodology, one of several offshoots of the ideas first proposed by Larry Constantine and augmented by Edward Yourdon. This particular branch was enhanced by Chris Gane, one-time Vice President at Yourdon, and Trish Sarson, also a past member of the Yourdon team, who are now the principals of Improved Systems Technologies, Inc., where they offer a system development methodology known as STRADIS, plus training seminars and workshops in its use.
- **HALSTEAD, MAURICE** The late Professor Maurice Halstead of Purdue University developed a system he called "software science" or "software metrics" for relating such things as program length, the time taken to code the program, and the expected number of errors to various measurable properties of the program. He established, for instance, by statistical analysis and measurement, that in a FORTRAN program, the average number of operands and operators in a source statement is 7.5, and that the equivalent measure for an assembler-language program is 3.7. A concise summary of Halstead's theories is given in Communications of the ACM, Vol. 23, No. 8, August 1980, pages 476-477.
- **HIPO - HIERARCHY PLUS INPUT PROCESS OUTPUT** HIPO is an IBM-originated program design-aid and documentation technique, part of a package

of tools dubbed IPT - Improved Programming Technologies - released by IBM in the early 1970s. HIPO is intended primarily as the documentation system in a top-down design of systems. It consists of a chart showing the hierarchy of processes in the system (the "H" chart); for each process, there is also an Input-Process-Output ("IPO") chart. Unlike other data-oriented structured design methodologies (e.g., Jackson, Warnier-Orr), HIPO is primarily process-oriented.

- **IMS - INFORMATION MANAGEMENT SYSTEM** IMS is the primary IBM data base management system for the 360/370, 303X and 4300 hardware series. It has an associated telecommunications monitor, generally called DC or IMS/DC, which interfaces the on-line terminals with the applications programs using IMS in much the same fashion as CICS does for its application programs.
- **INSTACODE** The marketing name for General Automation's hardware systems packaged to support the NOCODE menu-driven programming system.
- **INTERTEST - INTERACTIVE TEST CONTROLLER** INTERTEST is a program development aid especially tailored for operation with IBM CICS. When installed, INTERTEST is automatically invoked whenever the CICS supervisor attempts to pass control to any of its application programs. If the application program about to receive control is designated for monitoring by INTERTEST, the designated transactions, terminals or other program items and steps are monitored in a special area of memory. INTERTEST prevents monitored programs from making all but a designated set of modifications in the CICS partition. This permits a high degree of security for debugging new programs while other CICS production programs are running unaffected. INTERTEST is capable of verifying the reentrancy of programs by detecting and blocking self-modification. Automatic breakpoint control suspends the monitored program upon detection of an error, allowing the terminal operator to make corrections. Symbolic debugging of COBOL and assembler CICS programs is an option. INTERTEST is marketed by On Line Software International, 65 Route 4 East, River Edge, NJ 07661, (201) 488-7770. The basic package costs \$9,000; symbolic feature is \$5,000 additional.

- **IPT - IMPROVED PROGRAMMING TECHNOLOGIES** IPT refers to a group of productivity-improvement tools introduced by IBM in the early 1970s. The group includes the Chief Programmer Team, Structured Walk Throughs, HIPO and Pseudo-Code. See also under each of these headings.
- **JACKSON DESIGN METHODOLOGY** This program design methodology, developed by Michael Jackson-in England, is based on the structured programming constructs (sequence, iteration and selection) and takes as its departure point the design of data structures. Program structures are designed to match the data structures, with special provisions to resolve cases where such a match is impossible ("structure clashes"). A simple diagram system, consisting of boxes and lines, is used to represent both data and process structures. Based on Jackson's method, augmented by top-down implementation and structured walk-throughs, Exxon Corporation developed the PST (Program Structure Technology), supported by an interactive graphics package called PSTAIDS.
- **LCP/LCS** LCP (Logical Construction of Programs) and LCS (Logical Construction of Systems) are the original names of what is now commonly referred to as the Warnier-Orr Methodology (see Warnier Orr).
- **MAESTRO** Maestro is a standalone "programmer's workbench" system, developed by Softlab GmbH of Munich, Germany, to run on Four-Phase computers. Among the tools offered by the Maestro system are: "Structograms" for structured COBOL programming; HIPO charts; interactive construction of decision tables, with automatic checking of completeness, inconsistencies and redundancies; syntax guidance for COBOL programs; and the standard text-editing features, including searching, thumbing and marking. Programs developed under Maestro are generally intended to run on IBM 370/303X mainframes. The system was, until recently, marketed in the U.S. by ITEL; it is now marketed by Maestro Systems, 3 Embarcadero Center, San Francisco, CA 94111, (415) 986-1188. See also Programmer's Workbench.

- **MATRIX ORGANIZATION** An organizational scheme in which specific projects are staffed by personnel from functional departments. Project people continue to report to the functional department heads but perform project-oriented tasks for the project leader. Matrix organization is an attempt to combine the longer-term concerns that are addressed by a functional organization with the short-term, project-oriented goals of a pure project-organization. There are many variations of matrix organization. "Professional" project leaders may have business responsibility for a number of unrelated projects, while functional group managers, relieved of day-to-day project concerns, are free to concentrate on recruiting, training, standards and career development.
- **MENU-DRIVEN PROGRAMMING** A generic term describing systems in which some or all of the traditional procedure-oriented programming tasks are replaced by "fill-in-the-blank," menu-driven, interactive sessions with the user. Examples include the IBM DMS/VS and IMS/ADF systems (see entries). The recently announced NOCODE system by General Automation is another example of menu-driven coding system. Wang computers employ menu-driven programming to replace traditional JCL sequences. The term "nonprocedural programming" has been applied to menu-driven systems in some literature; INPUT prefers its terminology to avoid confusion with such nonprocedural languages as SIMSCRIPT.
- **NASSI-SCHNEIDERMAN CHARTS** Nassi-Schneiderman charts are a system of stylized flowcharts designed for use with structured programming. The system has graphical representations for the basic constructs of sequence, iteration and selection.
- **NCSS** In the context of software reliability and quality assurance, NCSS stands for Non-Commentary Source Statements. For example, the estimated cost of applying a given program testing methodology might be described in terms of dollars per NCSS.

- **NCSS** is also the abbreviation for National CSS, a computer services firm that supplies timesharing services in conjunction with its 3200 on-site hardware.
- **NOCODE** A menu-driven programming system developed for General Automation minicomputers ("Instacode system"). See Menu-Driven Programming.
- **PDL - PROGRAM DESIGN LANGUAGE** PDL is a high-level pseudo-code system designed to aid the production of structured, top-down designs. It combines plain English vocabulary with syntactical constructs representing the basic structured programming elements of iteration (DO) and selection (IF). A computer program is available to record system descriptions expressed in PDL and to produce reports, giving cross-references between program segments and using the standard indentation technique to show levels of nesting. PDL is close enough to actual code that such code can be produced from the PDL description quickly. PDL has no special facilities for describing data structures.
- **PETRI NETS** Petri nets are a formalized technique for the representation and analysis of complex, dynamic systems comprised of multiple, interrelated units operating simultaneously. This technique, devised by C.A. Petri of Germany, is particularly valuable for defining concurrency of parallel processes. It has been applied primarily to the description and analysis of operating systems and computer systems, but can also be used in the design and analysis of any large software system. Petri nets are directed graphs consisting of "places" (circles) which may contain "tokens"; places are linked by directed arcs; and "transitions" may bisect the links between places. The analysis of system behavior based on its Petri net representation can become very complex very quickly with increasing system complexity.
- **PRIDE-Logik/ASDM** This is a system design methodology marketed by Milt Bryce Associates of Cincinnati, OH. The degree of computerization varies from none in PRIDE itself (Profitable Information by Design) to some when

Logik is added (Logical Organization and Gathering of Information Knowledge); the combination is then called ASDM - Automated System Design Methodology. Additional computerization takes place with the addition of ADF - Automated Design Facility (a data dictionary manager) - and Genasys (a code generator). Key concepts of the system are: structured system and data component design; recognition of five levels of system components and six types of data components; system-data relationships; and a nine-phase design/development cycle. When fully configured, the system, which is written in COBOL, has its own data base, which is created from input forms furnished by the designer. It can produce a variety of reports, charts and documentation and perform checks for completeness and accuracy. The system runs on IBM 370, Burroughs, Honeywell, CDC, DEC, ICL and Univac mainframes and on Hewlett-Packard 3000 minicomputers.

- **PROGRAMMER'S WORKBENCH** This term is used both as a generic name for a class of on-line program development systems, and specifically for a particular implementation of the concept under the UNIX operating system for the DEC PDP-11 line of computers. PWB/UNIX was developed by Bell Laboratories to give programmers a variety of powerful on-line tools for program development, documentation and testing. The product is marketed "as is" (i.e., without ongoing support) by Western Electric, and is also licensed to resellers who add features and offer additional documentation and support. Other products that fall in this category include Maestro (see entry) and Programmer's Workstation, both of which run on Four-Phase equipment, but only the latter is marketed by Four-Phase. PWB/UNIX can be used to develop programs for UNIX, as well as for IBM and other "host" systems under an RJE arrangement. Maestro and PWS are intended specifically as standalone program development machines for IBM host systems.
- **PSEUDO-CODE** This is a generic term, describing a language that generally combines structured control constructs (IF-THEN-ELSE, DO WHILE) with free-form, natural-language English. It is also the name of a specific implementation, released as part of IBM's IPT (Improved Programming Technologies). The term Metacode has also been used for this type of

language, of which a prime example is PDL (see entry). The idea is to express the functions required at each step of a proposed program in natural English, while retaining the control (branching) mechanisms of the eventual target programming language. Pseudo-code is more readable than a COBOL or FORTRAN program, and at the same time can be translated into the target language very quickly.

- **PSL/PSA - PROBLEM STATEMENT LANGUAGE/PROBLEM STATEMENT ANALYZER** PSL is a language for describing information processing systems; PSA is a computer program that analyzes PSL descriptions and produces reports and messages. Both have been developed by Daniel Teichroew and Ernest A. Hershey, III of the ISDOS project at the University of Michigan. PSL is based on the model of an abstract system, consisting of objects, which may have properties, which may in turn have property values. The objects may be interconnected in various ways, which are called relationships. The PSA builds a data base from the system description expressed in PSL, and can produce a number of reports, including modification history, a list of all objects with their type and date of last change, a list of all properties and relationships for a particular object, and analyses of gaps in the information flow or unused data objects. A graphic (flow chart) description of the dynamic behavior of the system can also be produced automatically by PSA. PSA, which is coded primarily in FORTRAN, is operational on IBM 370, Univac 1100, Honeywell 600/6000 and PDP-10. The PSA software is available to "participants" in the U. of Michigan sponsored PSL/PSA research; "participation" costs \$15,000/year.
- **PST, PSTAIDS** See Jackson Design Methodology.
- **RELATIONAL DATA BASE** A data base and the associated Data Manipulation Language (DML) that permits the user to view the data as a set of relations (tables) and to operate on relations as wholes, rather than one record at a time.

- **REUSABLE CODE** Reusable Code is both a generic term, describing systems in which standardized functions are coded into universally available code modules, which are catalogued in a dictionary and filed in a data base; and a specific product, to be offered by Raytheon. The Raytheon Reusable Code System consists of four main elements: a logic shell generator, based on structured design; a reusable code library containing some 1,500 modules; a reusable code dictionary, describing the modules so that the desired ones can be located by keyword search; and a productivity measurement system. Kapur Associates of Danville, CA, is also marketing a reusable code methodology.
- **RSL/REVS** RSL (Requirements Statement Language) and REVS (Requirements Engineering and Validation System), developed by TRW for the Ballistic Missile Defense Advanced Technology Center, are very similar to PSL and PSA respectively (see PSL/PSA).
- **SADT - STRUCTURED ANALYSIS DESIGN TOOL** SADT is a proprietary, structured design methodology based on the structured analysis concepts of Douglas T. Ross. The system is being marketed by SofTech, 460 Totten Pond Rd., Waltham, MA 02154, 617/890-6900. SofTech does not market any computerized assist, but a tool is available on Cybernet. The SADT methodology employs a simple but highly disciplined graphic "language" to describe successive levels of action and of data. A key concept, forcing terseness and clarity, is that a "bound context," limits each "box" in the charts to dealing with a completely separate activity, related to the rest of the system by well-defined input(s), output(s) and control(s). Terseness and hierarchical clarity are also promoted by the rule that limits each page to no more than six "boxes" and each box or arrow label to no more than six words. A team with a clearly defined mission for each member ("authors," "readers," etc.) is also part of SADT; it is somewhat similar to the Chief Programmer Team from IBM. SADT is being promoted as an aid in system requirements and specifications, a system design discipline and a documentation system. The tool is especially popular in Europe.

- **SDM/70 - SYSTEMS DEVELOPMENT METHODOLOGY** SDM/70 is a commercial, manual system development methodology, consisting of six volumes covering the following topics: system life cycle and methods; documentation standards; estimation guidelines; administration, management and executive involvement guidelines; quality assurance; user involvement guidelines; and tutorials on such subjects as structured design, HIPO, data base management, etc. The system has been used for projects other than DP as well. Usage can start at any phase, for large, small and maintenance projects. The system has an interface to PC/70, an automated project control system from the same vendor. An enhanced "Version 2" has been recently released. The complete system sells for \$30,000; an abbreviated version lists for about \$10,000 less. The vendor is Atlantic Software Inc., Lafayette Building No. 910, 5th & Chestnut St., Philadelphia, PA 19106, (215) 922-7500.
- **SPECTRUM** SPECTRUM is one of several commercially available, completely manual system development methodologies. The SPECTRUM methodology is described in a set of from 18 to 25 "books," some of which are specialized to specific tasks, so that people performing these tasks (programmer, analyst, supervisor and project manager) need not consult the entire set of books. Three separate life cycles are defined for (a) major projects, (b) small projects, (c) maintenance. Procedures are also defined to control purchased software and to assure that maintenance activities are reflected in the documentation. Two versions are currently offered: the full package, SPECTRUM-1, costs \$42,000; a reduced version, SPECTRUM-2, costs about \$10,000 less. The supplier is Spectrum International Inc., 32107 West Lindero Rd., Westlake Village, CA 91361, (213) 991-5350.
- **SPF - STRUCTURED PROGRAMMING FACILITY/SYSTEM PRODUCTIVITY FACILITY** An optional feature for TSO, SPF is a support system for the utilization of 3270-type CRTs in TSO program development activities. Its name is somewhat misleading, in that its support for structured programming is essentially limited to easing the task of indenting source code. The majority of SPF features are simply intended to allow the utilization of the capabilities of the 3270 CRT, such as full-screen editing and function keys. A newer

version of SPF, which can run with either TSO or CMS, is now available from IBM; the initials of the "new" package stand for System Productivity Facility.

- **STRUCTURED ANALYSIS** A generic name for a group of top-down, hierarchical design methodologies, including those developed by Larry Constantine, Edward Yourdon, Tom DeMarco, Chris Gane and Trish Sarson, Victor Weinberg, Douglas Ross, Jean Dominique Warnier and Kenneth Orr.
- **STRUCTURED PROGRAMMING** A program coding discipline that, in its purest form, admits only three types of program constructs: sequence, iteration and selection (also called choice or decision). A mathematical proof that any program that can be represented by a conventional flow-chart can also be constructed exclusively from these three elements, was published in 1966 by Corrado Bohm and Giuseppe Jacopini (so-called Jacopini-Bohm or Structure Theorem). Edsger W. Dijkstra of Holland and Harlan Mills and R.W. Floyd of the U.S. have also made major, early contributions to the subject. Structured programs require no GO-TO statements; the technique was therefore initially known as "GO-TO-less programming". Limiting the branching logic to just two standard forms (iteration and selection) results in programs that can be read and understood strictly from the top down, without the "jumping around" that characterizes conventional, unrestricted-branching program. Structured programs are easier to maintain and modify. Because of its "top-down" property, structured programming is especially suitable for implementing top-down designs. Structured programming is now often combined with other techniques, such as hierarchical design, chief programmer team, and structured walk-throughs, to form various program design methodologies.
- **STRUCTURED TABLEAU** A structured system design method developed by Donald R. Franz of Southwestern Bell (St. Louis). The method is based on the use of classical decision tables to describe decisions and the corresponding actions in a succinct, unambiguous way. Franz augmented the classical decision table by adding the required structured design constructs for iteration and sequence, and called the resulting artifice "Structured Tableau" to

distinguish it from "plain" decision tables. The method offers a simple, quick way to assure design completeness and to estimate resulting program complexity.

- **STRUCTURED WALK-THROUGHS** A programming design review technique introduced by IBM as part of the Improved Programming Technologies package. Assumed to apply to Chief Programmer Team organizations, it can also be used with conventional programming organizations. The purpose of the technique is to eliminate the negative aspects normally associated with reviews. The developer (coder or designer) is responsible for calling the review, selecting the reviewers, setting the objectives and distributing copies of the work to be reviewed before the meeting. Four to six people and one to two hours are typical of good walk-throughs. The developer "walks" the reviewers through the work by explaining the work in a step-by-step manner, which simulates the actual execution of the program or function. This process brings to light errors in logic and coding. There is no relation between structured walk-throughs and structured programming, although the use of the latter eases the task of making sure that the walk-through has been complete.
- **SYSTEM DEVELOPMENT METHODOLOGY (SDM)** SDM is a set of ground rules for the development of systems, generally containing at least several, if not all, of the following elements: project control mechanism; guideline for system analysis and design (usually based on structured techniques); a well-defined, multiphase project life cycle definition; definition of deliverables at the conclusion of each phase and sign-off procedure, and procedures for iterating (implementing changes); documentation standards; estimation techniques; test plan; tutorials (written or computerized explanations on how to use the system). Some SDMs are entirely manual; i.e., they consist of written descriptions of procedures; for example, SDM/70, CARA and SPECTRUM. Other SDMs may be computerized to varying degrees; for example, PRIDE-Logik/ASDM with ADF, SYSTEMACS.
- **TAPS** Terminal Applications Processing System. See TTP.

- **TOP-DOWN DESIGN** A design discipline in which the design process is carried from the highest level of generality (abstraction) through successively more detailed levels, until the desired degree of detail (e.g., functions directly implementable in a given programming language) is reached. Each general function or task identified at the highest level is successively expanded into progressively more detailed or elementary functions. Functional decomposition is another name for top-down design methodology. In programming (coding), top-down implementation calls for coding the most general control routines first, leaving the implementation of "stubs" (calls to as-yet-uncoded subroutines) for later levels. In this way all calling sequences are defined before any of the called subroutines are coded, eliminating a major source of programming errors. The term "top-down" is also used to loosely describe a property of structured code: it is sequentially readable, from the top down, without the need to "jump around" as in more conventional coding.

- **TSO - TIME SHARING OPTION** TSO is a system for supporting multiple interactive terminals engaged in program development and testing activities. Originally available under IBM's MVT operating system, it is now found mostly in MVS systems. To the operating system, TSO appears to be just another user; but TSO acts as a telecommunications monitor and supervisor for its terminals, allocating system resources among these terminals. TSO includes text-editing facilities for the creation and maintenance of source code, and an interface to the language compilers to permit dynamic debugging.

- **TTP - TAPS TRANSACTION PROCESSOR** A software package intended to ease the task of developing transaction-oriented application programs. The package includes a communications monitor similar to CICS or IMS/DC (the latter two can be substituted for the native CM). Like CICS and IMS/DC, TTP also has data base management facilities, transaction processing, and message handling/CRT screen support. TTP also incorporates menu-driven programming features, like those of DMS. Unlike CICS or IMS, which run only on IBM mainframes, TTP runs on both IBM machines (under OS and DOS) and on a number of minicomputers, including Interdata, DEC, HP, Prime and Harris. An interface to Intel/MRI System 2000 DBMS was announced in July, 1980. It

is marketed by Decision Strategy Corp., 708 Third Ave., New York, NY 10017, (212) 687-6548 (recently acquired by Informatics, Inc.). Pricing ranges from \$25,000 for a typical minicomputer, to \$30,000 under IBM DOS and \$40,000 under IBM OS.

- **WARNIER-ORR** A system and program design method based on concepts developed by Jean-Dominique Warnier in France and Kenneth Orr in the U.S. The system emphasizes structured data design as the starting point. Data structures are expressed in the Warnier Data Diagram. From the data structure, using time sequence analysis and change analysis, W/O derives the procedural program design almost as a one-to-one translation. The control structures for the program are first translated into pseudo-code, and then into actual code. A computerized system for creating Warnier-Orr diagrams, called STRUCTURES(S), as well as seminars on the W/O methodology, are offered by Langston, Kitch and Associates, 715 East 8th St., Topeka, Kansas 66607.
- **YOURDON, EDWARD** Author of a structured analysis methodology. See Constantine/DeMarco/Yourdon.

APPENDIX D: STRATEGY FOR A TRAINEE-RICH ORGANIZATION

APPENDIX D: STRATEGY FOR A TRAINEE-RICH ORGANIZATION

A. THE PROBLEM

- A data processing manager is looking for a new technical employee and has a choice between candidates A or B.
- A's qualities:
 - Temperamental.
 - Overpriced.
 - Obsolescent skills.
 - "Know-it-all," closed to new approaches.
 - Limited potential for advancement.
 - Little loyalty to the firm.
 - Not disposed to work for you.
- B's qualities:
 - No ego problems.

- Underpriced.
 - No bad habits technically.
 - Open to new ideas and techniques.
 - Potential for several promotions.
 - Great loyalty to firm.
 - Eager to work for you.
- The manager will usually choose A. Why? Because A is an experienced programmer, while B is a trainee. Trainees are viewed as a last resort.

B. THE OPTIONS

- Many authorities project an increasing scarcity of experienced technical staff through the 1980s. This will put increasing pressure on EDP management to find alternatives to competing for the same pool of experienced staff. The main options are to:
 - Rely less on technical staff through the use of "user-friendly" systems.
 - Decrease turnover.
 - Upgrade existing EDP staff (e.g., operations staff to programmers).
 - Transfer non-EDP staff to the EDP department.
 - Hire trainees from the outside.

- The first option is problematic and not under the control of the typical EDP staff. Decreasing turnover is a popular, but elusive, goal; in any event it will not help installations that are expanding. The last three options all involve trainees in one form or another.

C. THE REASON FOR THE PROBLEM

- Why isn't the use of trainees more popular? A trainee:
 - Can't start programming Monday.
 - Requires training overhead (the trainee's own time, plus, more importantly, that of experienced staff).
 - May have no interest in or potential for the work.
 - May go somewhere else as soon as training is over.
- These are real issues (although there also appears to be a substrata of irrational prejudice). Dealing effectively with the first two issues requires a planning process that is adhered to. Weeding out those with no interest or aptitude means avoiding putting trainees right off the street into a programming course or programming job. Methods to avoid this include:
 - Hiring graduates of two-year data processing programs or commercial programming schools.
 - "Apprenticing" some technical trainees initially to operations areas.
 - Encouraging internal company transferees to work on short-term duty in the EDP department before entering the training program.

- The problem of "training a programmer for someone else" is the heart of the rational opposition to using trainees.
 - Trainees will leave, but for the same reasons others leave:
 - Lack of opportunity to learn or to work on challenging projects.
 - A nonsupportive management structure.
 - Lack of technology.
- On the positive side, trainees are very conscious of the career opportunity given to them by their employer (as well as to the resources invested in them).
 - In addition, normal human inertia is a reason not to leave.
 - From the trainee's standpoint, what do they see?
 - Depending on the size of the firm, on the average one-half to two-thirds of the staff brought into the firm have previous experience. Other things being equal, most of these jobs could be filled by promoted staff, with new hires brought in at the trainee level.
 - The practice of hiring experienced people from the outside sends subconscious signals to trainees (as well as to many other employees) that their employer does not value or trust them.
 - Therefore, from the employee's standpoint, it is the company that first broke faith and has been disloyal.
- Why do firms go outside, rather than promote from within?

- For the most part it represents a planning breakdown.
 - . New projects are "suddenly" approved.
 - . The right person is in the middle of another project.
 - . Available candidates are "too junior."
- Planning "breakdown" is often a euphemism for no personnel planning at all. (That is, no detailed lists or projections of each person's skills and the needs of the organization).
- In addition, there is the "grass-is-greener" syndrome: The present staff's weaknesses are known, while the qualities of newcomers can be fantasized.

D. THE SOLUTION: A TRAINEE-RICH PERSONNEL STRATEGY

- What is needed is a way of introducing trainees into the organization in a way that trainees can be highly useful as soon as possible.
 - Turnover can then be controlled by management rather than by the individual.
- The first point represents a reversal from current practice. Currently, EDP management does what it can to retain all programmer/analysts, except the obvious losers. This leaves management at the mercy of each individual's personal plans and preferences.
- Establishing a trainee-rich organization is accomplished by:

- Building into personnel projections a 25% attrition rate for the first three years, most of which will come from management initiative.
 - Selecting trainees carefully, giving preference to existing staff, two-year data processing graduates and general college graduates, rather than computer science majors.
 - Telling trainees in advance about the attrition policy, but assuring them that those who cannot be retained will nevertheless have received quality training that will make them very desirable on the job market.
 - Assigning most trainees for their initial 6-12 months to the operations area (with released time for programming training) so that both the trainee and the manager can make up their minds whether to commit to further training.
- For the initial year of programmer training, assign each trainee to a second-year trainee since:
 - The mentor can identify easily with the first-year trainee.
 - Teaching is the best way to learn.
 - The second-year trainee's supervisory skills can thus be gauged and improved.
 - Certain trainees would be "fast-tracked" during their second year. These individuals would get additional responsibilities, and would be less likely to want to leave. (They would probably also be too "junior" to be attractive elsewhere.)
 - Other trainees would be told at the end of their second year that, while they were good, there were others who were even better and therefore the company would assist them in finding desirable jobs.

E. STRATEGY REQUIREMENTS

- The most important requirement is that there be a philosophy and commitment to promotion from within. If this is not the case, then:
 - Trainees and ex-trainees will have no assurance that there is a future for them with the company.
 - There will, in fact, often not be desirable jobs for internally grown staff. This will be a special problem for those in the "fast track."
 - Trainees will have less loyalty to the firm, since it displays less loyalty to them.
- To make possible promotions from within will require the EDP department to have a reasonably precise projection of its personnel needs, including skill levels. This should be mirrored by an inventory of present and planned skills in the staff. It is often the lack of such planning and information that causes firms to recruit experienced staff from the outside.
- But what if there is a new "rush" project, or there just aren't the right in-house skills available, or they can't be developed in time?
 - Outside consultants should be used to bridge personnel shortages. In a trainee organization there will be far less resentment of the consultant's higher salary, since everyone else makes more money than trainees anyway. The trainees will also be able to learn from these more experienced people. (Care must be taken, though, that the consultant has good work habits!)
 - The lack-of-skills issue can be turned to the organization's benefit by hiring a consultant, part of whose duties would be to train the firm's staff in the skill area needed.

- Some organizations object to the use of consultants over long periods of time because of the cost involved. This is a valid objection where quality of the consultant's work is not commensurate with costs.
 - A way around this would be to hire experienced programmers who cannot work 9-5 because of family responsibilities; their motivation and skills are often very high, and they are willing to work for a reasonable salary.
- This last point leads to another key requirement: an effective training program. In all too many organizations, training is an episodic thing, with trainees initially exposed to a block of classroom training followed by a largely unstructured on-the-job experience, too often consisting of picking up bad habits from fellow employees.
- The defect of most training programs is that the initial block of education provides far more information than most students can actually put to use. A better approach would be to:
 - Put trainees to work immediately in the operations area or, possibly, as a project librarian.
 - Concurrently, on a released-time basis, have them go through modules of in-house video tape education courses, supplemented by live lectures from existing staff. This will keep existing staff on their toes, while helping to ensure that approved methods are actually followed.
 - Trainees would progress to coding assignments at their own rate; i.e., there would be considerable incentive to learn as fast as possible.
 - Once assigned to a programming group, released-time education would continue for trainees and, to a lesser extent, for all staff.

- The main method of learning reinforcement would not be didactic, classroom-type training, but tutoring and peer training:
 - Peer review via such techniques as chief programmer teams and structured walk-throughs would share information and root out inferior approaches.
 - The "quality circle" concept should be part of the organization's approach.
 - Measurement tools (e.g., bugs per 1,000 lines) would have a role, but largely as a feedback device for the group.
- The objection to the concept that a higher level of management skill is needed, especially at the two lowest supervisory levels, is not valid. In fact, the skill levels needed are no higher than what should be minimally acceptable anyway. If there are problems, a new management development program or new managers are needed.
- The objection that an organization does not get "new blood" from trainees is fallacious. Middle-level hirings usually don't bring in many new ideas or a healthily critical attitude. If this is what is meant by "new blood," then the judicious use of capable consultants is what is needed.

F. ORGANIZATIONAL IMPACT

- There is little question that the cumulative effect of implementing a trainee-rich approach would be to change the EDP organization in significant ways:
 - The amount and effect of planning would be increased.

- Training would be continuous, creating a more questioning organization that would try new approaches (not all of which would work).
 - There would be a greater group consciousness, although the people themselves would be more heterogeneous (i.e., there would be equal employment advantages).
 - The average age of the organization would decrease until equilibrium was reached. Energy levels are likely to increase.
 - Skill levels, morale and productivity would increase. The turnover percentages themselves would probably not decrease markedly, although the negative impact on the organization would be far less.
- For most organizations, the net effect would be positive, although some of the long-time employees might have difficulty adjusting. At this point, management will have to decide where their skills are most valuable, and act accordingly.

APPENDIX E: SELF-ANALYSIS, ACTIVITY-ORIENTED,
PRODUCTIVITY AUDIT

APPENDIX E: SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT

- This appendix is designed to allow data processing management to rate themselves and their organizations in the following areas of potential productivity improvement. These areas are considered largely under the control of EDP management:
 - Organization and personnel policies and practices.
 - Internal control and planning.
 - User relations.
 - Programming support.
 - Productivity tools.
- These self-analyses are designed so that each organization can rate itself in each of the different areas. Please note that, in some cases, different point scores are possible, depending on an individual firm's stage of EDP development.
- Some initiatives are only appropriate at certain stages. Explanations and comments for ratings in a particular area are supplied under the heading "Rationale."

- Select only one point rating for each topic headed by a hyphen; total the positive and negative points separately.
- The range of maximum ratings, as shown in Exhibit E-1 extends from 84 negative points to 146 positive points for organizations in stage zero, increasing to a range of 103 negative points to 181 positive points for organizations in stage four. Both risks and rewards are higher in the later stages.

TOTAL RATING RANGE FOR SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT BY STAGE

AREA	STAGE 0	STAGE 1	STAGE 2	STAGE 3	STAGE 4
● ORGANIZATION AND PERSONNEL POLICIES AND PRACTICES					
- MAXIMUM POSITIVE POINTS	48	50	51	51	51
- MAXIMUM NEGATIVE POINTS	31	31	33	33	33
● INTERNAL CONTROL AND PLANNING					
- MAXIMUM POSITIVE POINTS	20	33	34	34	34
- MAXIMUM NEGATIVE POINTS	12	19	20	20	20
● USER RELATIONS					
- MAXIMUM POSITIVE POINTS	17	17	17	17	17
- MAXIMUM NEGATIVE POINTS	2	2	4	10	10
● PROGRAMMING SUPPORT					
- MAXIMUM POSITIVE POINTS	8	8	8	9	8
- MAXIMUM NEGATIVE POINTS	7	7	7	7	7
● PRODUCTIVITY TOOLS					
- MAXIMUM POSITIVE POINTS	53	55	71	71	77
- MAXIMUM NEGATIVE POINTS	32	33	39	39	39
● TOTAL					
- MAXIMUM POSITIVE POINTS	146	163	181	181	181
- MAXIMUM NEGATIVE POINTS	84	92	103	103	103

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<u>I. Organization and Personnel</u>			
<ul style="list-style-type: none"> ● Organization. <ul style="list-style-type: none"> - Organizational structure is: <ul style="list-style-type: none"> • Mature, appropriate to the job to be done, and may involve matrix, matrix detailing, project under user manager, under technical manager, or distributed structure; accompanied by strong sense of mission, strong management; considered a good model within the industry; in place over four years. • Evolving as above, but less than four years' experience. • Project-oriented, staff rotated among business functions approximately every two years; specialist advisors participate on projects and perform testing integration, etc. • Project-oriented, but no conscious rotation of staff; specialized teams exist for testing, integration etc. 	<p>+3</p> <p>+5</p> <p>+6</p>	<p>0</p> <p>1</p> <p>2-4</p>	<p>By definition, an organization in Stage 0 cannot be very effective.</p> <p>Organizations in Stage 1 are not as able to effectively include users as those in later stages.</p>
	+3	0	As above.
	+4	1-4	
	+3	0-4	An adequate approach in lower stages that becomes less adequate in higher stages.
	+2	0-1	
	-1	2-4	This approach becomes a disability in later stages because it offers limited flexibility and breadth of experience.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Business function-oriented, exposure to other functions available only via permanent transfer. 	-1 -3	0-1 2-4	This becomes a serious disability in later stages.
<ul style="list-style-type: none"> Maintenance and new development: <ul style="list-style-type: none"> Are separated, both managed as projects (scheduled-release basis). Are separated, maintenance managed as a priority queue. Are combined, duties rotated among all staff. Are handled <u>ad hoc</u>; maintenance predominates by more than 2:1 ratio. 	+3 +2 +1 -2	0-4 0-4 0-4 0-4	This represents a continuum that applies to any stage of development, although it is unlikely that the higher-value modes will be found in organizations at stages 0-1.
<ul style="list-style-type: none"> Objectives and Motivations. <ul style="list-style-type: none"> Objectives: <ul style="list-style-type: none"> Are clearly articulated and strongly espoused by all levels of the organization; EDP department objectives align with and support corporate objectives; productivity improvement designated as an objective. 	+5	0-4	This represents a continuum that is applicable to organizations at any stage. The high maximum point value reflects the unifying influence of each individual's understanding how his or her objectives relate to those of the organization.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> As above, but known and apply to top levels only. Are not clearly stated, but are being discussed. Are unknown. Are "not applicable to EDP." 	+3 +1 -1 -3	0-4 0-4 0-4 0-4	
<ul style="list-style-type: none"> Motivations: <ul style="list-style-type: none"> Center around a personal, project and departmental sense of value and contribution. Come primarily from management or peer recognition of accomplishment. Come from individuals understanding where they stand. Are related to salary, advancement and learning opportunities. Vary from group to group, but are being addressed as an improvement objective. 	+6 +5 +4 +3 +2	0-4 0-4 0-4 0-4 0-4	This represents a continuum applicable to any stage of development, and reflects INPUT's survey research findings that indicate the relative values assigned by individuals to their own motivating factors.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Are left to each individual. Provide negative reinforcement, but no positive reinforcement. Are seen by some staff as favoritism. 	<p>-2</p> <p>-4</p> <p>-6</p>	<p>0-4</p> <p>0-4</p> <p>0-4</p>	
<ul style="list-style-type: none"> Corporate Policies. <ul style="list-style-type: none"> Corporate policies provide: <ul style="list-style-type: none"> Sufficient flexibility for flextime, part-time, work at home; adequate salary and advancement possibilities for non-EDP and EDP departments. Some of the above, but limited to EDP department. None of the above. Deficient salary and advancement possibilities compared to other companies in the area. 	<p>+3</p> <p>+1</p> <p>-1</p> <p>-3</p>	<p>0-4</p> <p>0-4</p> <p>0-4</p> <p>0-4</p>	<p>This represents a continuum applicable to any stage of development.</p> <p>May be seen as favoritism.</p> <p>Serious exposure in the long term.</p>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE		EXPLANATION
<ul style="list-style-type: none"> ● Recruiting and Staff. <ul style="list-style-type: none"> - Experience-level distribution of staff: <ul style="list-style-type: none"> • Thirty percent over five years in EDP with company, 50% have two to five years in EDP, 20% under two years. • More than 60% over five years, or less than 10% over five years. • More than 40% under two years, or less than 10% under two years. 	<p>+3</p> <p>+1</p> <p>-1</p>	<p>0-4</p> <p>0-4</p> <p>0-4</p>		<p>Good distribution of experience.</p> <p>Too many or too few "old timers."</p> <p>Too many or too few entry-level personnel, unless conscious attempt to develop "trainee-rich" approach. If so, +1. If in place three years or more, +2.</p>
<ul style="list-style-type: none"> - Personnel needs projected; recruiting and training solutions for next five years are: <ul style="list-style-type: none"> • In place (responsibility defined, written plan exists and reviewed annually, funding levels established). • In process of being implemented. 	<p>+5</p> <p>+3</p>	<p>0-4</p> <p>0-4</p>		<p>This represents a continuum applicable to any stage of development.</p>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Identified as action item for 1981. Not yet addressed. 	+1 -1	0-4 0-4	
<ul style="list-style-type: none"> In-house training and recruiting facility exists and provides 30% or more of forecasted requirements from internal resources. 	+2	0-4	<u>Bonus points:</u> represents a strong commitment to training; also enables organization to withstand fluctuations in personnel supply more effectively.
<ul style="list-style-type: none"> Turnover. <ul style="list-style-type: none"> Turnover statistics for EDP department: <ul style="list-style-type: none"> Stabilized within 5-10% range, including transfers; ongoing projects not seriously affected. Fluctuating over 10%, or stable but under 3%; some ongoing projects affected. Exceeds 15% range; most projects seriously affected. 	+3 +1 -2	0-4 0-4 0-4	Optimum, allows for new blood in the department. Either somewhat excessive or does not allow enough new blood; equally harmful. Excessive, destructive turnover level.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> - Vacant positions: <ul style="list-style-type: none"> . Do not exceed 5% of total staff. . Are 5-10% of total staff. . Exceed 10% of total staff. 	+5 -1 -3	0-4 0-4 0-4	Represents a continuum; inability to fill vacancies shows a negative judgment by the personnel market of the desirability of employment here, or unrealistic budgeting and planning practices.
<ul style="list-style-type: none"> • Career Development. <ul style="list-style-type: none"> - Career development plan: <ul style="list-style-type: none"> . In place for staff and managers for at least three years provides specific experience and training objectives considered a noncancellable priority; reviewed at least annually; provides separate managerial and technical paths. . As above, but in place for one to two years. . As above, but covers only managerial or technical path, or only some staff and managers. 	+5 +4 +3	0-4 0-4 0-4	This represents a continuum applicable to any stage of development.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> As above, but frequently preempted by other priorities. Available upon initiative of individual. Provides technical training only, not a career path. Under development. Does not exist. 	+1 +1 +1 -1 -4	0-4 0-4 0-4 0-4 0-4	
<ul style="list-style-type: none"> Morale. <ul style="list-style-type: none"> Morale and attitudes of staff and management are: <ul style="list-style-type: none"> Enthusiastic, confident; staff feels challenged and supported; strong sense of mission; the organization is widely known as a good place to work; quality circles or equivalent self-determination groups exist; staff and managers are publicly commended. Healthy, but less than enthusiastic; have changed noticeably down from enthusiastic, or up from griping, in last three to six months. 	+5 +2	0-4 0-4	This represents a continuum applicable to any stage, and reflects INPUT's observations as to relative values of each description.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Characterized by some griping, but performance measurements are stable; no strong sense of mission; feedback on personal accomplishments varies by project group. 	-1	0-4	
<ul style="list-style-type: none"> Sometimes discussed as a problem area; transfers are increasing; staff lackadaisical; feedback mainly negative. 	-3	0-4	
<ul style="list-style-type: none"> Openly discussed as a problem area; work output and quality clearly suffering; management attention directed to schedules or budgets, but not to "people" issues. 	-5	0-4	

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<p>2. Internal Control</p> <ul style="list-style-type: none"> Planning. <ul style="list-style-type: none"> The planning function: <ul style="list-style-type: none"> Is a defined and staffed full-time function, and includes personnel, hardware, software, communications and applications planning responsibilities; includes strong user representation; incorporates several levels of steering committee; functions on basis of a published year-round schedule. As above, but one level of steering committee; or does not include full set of responsibilities; or functions only at budget time. As above, but no steering committee; or user involvement is weak; or covers only hardware and communications responsibilities. 	<p>-1 -2 +3 +5</p>	<p>0 1 2 3-4</p>	<p>Inappropriate: "overkill." Inappropriate because emphasis on establishing basic controls. Appropriate, but full-scale implementation would be wasted. Appropriate level of activity.</p> <p>Inappropriate level of activity. Appropriate, but full-scale implementation would be wasted. Appropriate for this stage. Insufficient planning for this stage. Somewhat inadequate planning for this stage.</p> <p>Appropriate level of activity for this stage. Insufficient planning for this stage. Inadequate planning for this stage. Seriously inadequate planning for this stage.</p>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE		EXPLANATION
<ul style="list-style-type: none"> Covers hardware responsibility only; or not a full-time staff function; but produces a written plan. Less than above. Not seen by management as an essential, funded and staffed function. 	+1 -2 -4 0 -3 -4 -2 -5	0 1-2 3-4 0 1-2 3-4 0 1-5	Barely adequate. Insufficient planning at these stages. Grossly inadequate planning at these stages. The expected situation in "chaos." Inadequate at these stages. Totally inadequate at these stages. Even for "chaos," this is unacceptable. Doubtful if an organization can advance beyond stage zero with this view.	
<ul style="list-style-type: none"> Priorities and Measurements. <ul style="list-style-type: none"> Priorities: <ul style="list-style-type: none"> Are established on the basis of management of an applications portfolio; no more than 15% of total effort justified by "policy" rather than "cost/benefit;" a facility exists for users to service their own needs too small to meet overall priority thresholds. 	0 +2 +8	0 1-2 3-4	Would be pointless to attempt in this stage. Very doubtful if this approach would be beneficial in these stages. Very appropriate at these stages.	

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> As above, but more than 15% of total effort justified by "policy" rather than "cost/benefit" reason. 	0 +2 +7	0 1-2 3-4	Same as above, but more serious effort is required to define benefit at stages 3-4.
<ul style="list-style-type: none"> As above, but no facilities for user to service small needs. 	0 +2 +5	0 1-2 3-4	Same as above, but for later stages this inhibits potentially valuable experimentation by the user.
<ul style="list-style-type: none"> Are established by the users for each user department or division, and by a corporate-level steering committee for the corporation, which authorizes overall funding levels; committees are not "rubber stamps"; priorities cover new development, major enhancements, and planned maintenance, plus an EDP research and development allocation. 	+4	0-4	A useful approach in all stages, not as valuable as applications portfolio management for organizations in stage 3-4.
<ul style="list-style-type: none"> As above, but priorities do not cover maintenance, major enhancements or EDP research and development. 	+2	0-4	A useful, but weak, approach for all stages that ignores half the EDP expenditures.
<ul style="list-style-type: none"> Are established via funding or head-count levels by the user departments or divisions, but no corporate-level steering committee exists. 	+2 +1 -2	0-1 2 3-4	An appropriate strategy. A marginal strategy. Insufficient at these stages.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Are established by any method, but thereafter not shuffled by outside pressure. Are frequently shuffled by outside pressure. Are arbitrary or nonexistent. 	+1 -2 0 -5 -2 -5	0-2 3-4 0 1-4 0 1-4	<p>A marginal strategy that is at least predictable. Insufficient at these stages.</p> <p>Expected in this stage. Doubtful if an organization can advance beyond zero with this happening.</p> <p>Even "chaos" must have plans to change. Doubtful if an organization can advance beyond zero without planning.</p>
<ul style="list-style-type: none"> Measurements: <ul style="list-style-type: none"> Are objective; relate to controllable factors; are considered essential by management and staff; are periodically analyzed and reported outside the EDP department; relate to corporate objectives; have been in place more than three years; have been the basis for implemented improvements. Include at least five of the above characteristics. Include at least three of the above characteristics. Are objective; relate to controllable factors. 	+5 +4 +3 +2	0-4 0-4 0-4 0-4	<p>Represents a continuum applicable to any stage in theory. In practice the higher ratings are not usually found before stage three (except for computer operations), and can be counter-productive if taken too seriously at the early stages. This audit is suggested as a more appropriate substitute for measurement in the early stages, and a useful supplement to measurement in the later stages. See the chapter on measurements for more information.</p>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Are being collected for future analysis. Are not being collected. Are considered a joke, do not relate to controllable factors. Are considered harassment, are manipulated by staff or management. Are used punitively, focus on individuals rather than systems. 	<div>+1</div> <div>-1</div> <div>-2</div> <div>-3</div> <div>-4</div>	<div>0-4</div> <div>0-4</div> <div>0-4</div> <div>0-4</div> <div>0-4</div>	
<ul style="list-style-type: none"> Measurements (in addition to any positive rating above): <ul style="list-style-type: none"> Address maintenance issues. Are used as part of a comprehensive set of controls. Have been tested and proven valid in the organization's context. 	<div>+1</div> <div>+1</div> <div>+2</div>	<div>0-4</div> <div>0-4</div> <div>0-4</div>	Represent "bonus" points because rarely included in a measurement program; any or all may be added to positive scores above.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE		EXPLANATION
<ul style="list-style-type: none"> ● Internal Controls. <ul style="list-style-type: none"> - The internal control mechanism: <ul style="list-style-type: none"> • Is based upon a well-defined, annually reviewed management-by-objectives program; incorporates quantitative as well as qualitative measures; provides authority commensurate with responsibility functions by permission and example; rewards productivity; tracks and balances the variety of dimensions comprising the EDP task, as defined in the written description; is itself analyzed and modified based on experience gained. • Is missing no more than three characteristics defined above. • Is missing four to five characteristics defined above. • Is limited to either qualitative or quantitative data alone. 	<ul style="list-style-type: none"> -1 +8 -1 +5 +1 +2 +1 -2 	<ul style="list-style-type: none"> 0 1-4 0 1-4 0 1-4 0-1 2-4 	<ul style="list-style-type: none"> Would be counterproductive and frustrating. Vital to a smooth-functioning and effective DP organization. As above. Unusual for this stage, suggests organization may already be entering Stage I. A weakness for these stages that should improve with experience. Normal for these stages, if it exists at all. A serious deficiency. 	

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Is limited to a standard, single remedy (e.g., over-time, add more bodies, etc.). Is arbitrary, hasty, vacillating or punitive. Does not provide feedback to revise measures or estimates based on experience. 	0	0	Weak, even for "chaos."
	-3	1-4	A very serious deficiency.
	-2	0-1	Reinforces chaos.
	-3	2	A serious deficiency.
	-5	3-4	Doubtful if an organization can advance to these stages with this type of control.
	-3	0-1	Reinforces chaos.
<ul style="list-style-type: none"> Reporting Accomplishments. Is viewed in the total perspective of the EDP department's charter and objectives as a proper function; produces a regularly published (at least semiannually) reconciliation of goals and achievements; forms the foundation of next year's plan; credits accomplishments to the individuals responsible for achieving them; management accepts responsibility for failures. 	-4	2	A very serious deficiency.
	-5	3-4	Doubtful if any organization can advance to these stages with this type of control.
	+4	0-4	Represents a continuum applicable to any stage, but higher ratings more normal as well as more essential from stage two onward.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> As above, missing individual credits. Limited to accomplishments, not reconciled to goals. Not done. 	+2 +1 -1	0-4 0-4 0-4	
3. <u>User Relations</u> <ul style="list-style-type: none"> <ul style="list-style-type: none"> Involvement. User community involvement: <ul style="list-style-type: none"> Spans the life cycle from feasibility to replacement of a system; emphasizes cooperative interaction; focuses on business requirements, not automation of current procedures; is based on sufficient commonality of understanding to minimize misdirection; progressively assumes primary responsibility for system. Is most active at definition of requirements and acceptance of completed system, but is available at other times; focuses on business requirements. 	+3 NA +5 +3 +2	0 1 2-4 0-2 3-4	Desirable, but unlikely to be a true commonality of understanding. By definition, the "control" stage has little or no user involvement; if it exists, stage may be mistakenly categorized. Very important in later stages. This is a desirable situation in the early stages (but unlikely in stage 1). This is only marginally acceptable in later stages.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Is most noticeable at definition of requirements; not considered a critical function. 	+1	0-2	Very marginal, even in "chaos"; acceptable involvement in the "control" stage since even this little contact is higher than average.
<ul style="list-style-type: none"> Requires coordination of three or more diverse functions and/or significant geographical dispersion of users. 	-3	3-4	A serious deficiency in the later stages.
	-3	0-1	<u>Penalty Points:</u> Coordination with users is exceptional at these stages, and will likely make things worse.
	-2 from above	2-4	<u>Penalty Points:</u> Coordination between EDP and users is a key factor in making progress, and these constraints will certainly inhibit productivity, despite the best intentions.
<ul style="list-style-type: none"> Responsibility. <ul style="list-style-type: none"> User community responsibility: <ul style="list-style-type: none"> Comprises priority setting, definition of systems, preparation and supervision of testing, preparation of user manual, performance of routine maintenance and minor enhancements, and all retrieval activities. Comprises first four of above functions, plus most retrieval activities. 	+5	0	Highly unlikely that all users could do this in "chaos"; review description of stages.
	+5	1	It would be unusual for this to occur in the "control" stage; review description of stages.
	+8	2-4	A key factor in achieving higher stages.
	+3	0	Same as above.
	+3	1	
	+5	2-4	

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Comprises first three of above functions, plus some retrieval activities. Comprises definition of requirements, some testing and retrieval activities. Progress being made toward above objectives. 	+2 +2 +3 +1 NA +3 +1 -3	0 1 2-4 0-4 0 1 2 3-4	Same as above. This is a significant achievement in Stages 0 and 1, but marginal performance in 2-4. This is an indication that the next stage is achievable. This should be a warning sign - progress should already have been made. It is doubtful if the higher stages could be attained if actual progress had not been made.
<ul style="list-style-type: none"> User Satisfaction Levels: <ul style="list-style-type: none"> Are best described as "enthusiastic and supportive"; are regularly surveyed to maintain and improve level of service. Are best described as "cordial and cooperative"; are regularly surveyed to improve level of service. Are described as "adequate"; are regularly surveyed to improve level of service. 	+4 +3 +2	0-4 0-4 0-4	This represents a continuum applicable to any stage, and is a key "survival indicator" for the DP manager.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> - Physical facilities: <ul style="list-style-type: none"> . Designed for programmer/analyst use, provide adequate workspace and some degree of privacy; conference facilities scheduled less than one day in advance. . Provide adequate workspace and some degree of privacy; conference facilities limited. . Provide limited workspace (desk only); privacy difficult. . RJE, printer or terminal facilities not nearby. . Are overcrowded, noisy, open bull-pen style. 	<div>+3</div> <div>+1</div> <div>-1</div> <div>-3</div> <div>-5</div>	<div>0-4</div> <div>0-4</div> <div>0-4</div> <div>0-4</div> <div>0-4</div>	<p>This represents a continuum that has more serious implications when the values are negative than when they are positive. The higher values are unlikely at stage zero, and if found would point to problems in management's understanding of what productivity is all about.</p>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<p><u>5. Productivity Tools</u></p> <ul style="list-style-type: none"> ● Standards and Methodology. <ul style="list-style-type: none"> - Standards: <ul style="list-style-type: none"> • Have been in place for more than three years; are reviewed annually, with provision for interim modifications; are approved by a representative body from all governed units; comprise process as well as product standards; allow innovation within pre-scribed limits; provide levels of definition corresponding to extent of interaction of program, system, etc.; are the basis for periodic audit of existing systems; are widely considered helpful, not burdensome; are taught in new-hire training classes at all levels; are cross-indexed and accessible. • For each missing feature defined above, subtract one point; minimum score for no standards at all. 	<p>+5</p>	<p>0-4</p>	<p>This represents a continuum that evolves over time, and reaches the higher point values only after results are proven by experience.</p>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<p>- Methodology for systems development (also known as phased plan or life cycle plan):</p> <ul style="list-style-type: none"> Has been in place for three years; covers all development or enhancement systems above a certain level of effort, criticality or complexity; has a modified applicability below that level; defines phases, process, deliverables and responsibility; is integrated with the project control system. As above, but less than three years old. As above, but applies only to new development, or does not have a modified version for less-significant systems. As above, but not integrated with the project control system. Defines phases and deliverables, but not process and/or responsibility. Exists, but is not universally used. Does not exist. 	<p>+5</p> <p>+4</p> <p>+3</p> <p>+2</p> <p>+1</p> <p>-3</p> <p>-5</p>	<p>0-4</p> <p>0-4</p> <p>0-4</p> <p>0-4</p> <p>0-4</p> <p>0-4</p> <p>0-4</p>	<p>This represents a continuum that forms the foundation for almost all other tools. It therefore should be initiated as soon as possible because the results are only seen two to three years later.</p>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> ● Project Size and Estimation. <ul style="list-style-type: none"> - Size and time estimates for projects: <ul style="list-style-type: none"> • Are at least 95% accurate 95% of the time, across all types and sizes of project; are developed on a formula basis; are included as a factor in a productivity index. • Are 95% accurate 90% of the time, for the most common size range and type of projects; exceptions do not exceed 10% underestimated or 20% overestimated for any project; reasons for variations are analyzed and incorporated in future estimates; attempts are being made to develop or revise a formula basis for estimating; results are included as a factor in a productivity index. • Same as above, but variations of -20%, +30%. • Are 100% accurate 100% of the time. • Are 80% accurate 80% of the time; variations are analyzed and included in future estimates; results considered in evaluating productivity. 	<div>+5</div> <div>+4</div> <div>+3</div> <div>+3</div> <div>+2</div>	<div>0-4</div> <div>0-4</div> <div>0-4</div> <div>0-4</div> <div>0-4</div>	<div>This represents a continuum.</div> <div>If accuracy of estimates is not used in evaluating productivity, subtract one point from stated rating. Higher point values (+4, +5) are unlikely in stages 0-2.</div> <div></div> <div>Probably estimating too generously.</div> <div>A good achievement for most organizations.</div>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> - If sizes or projects are broken into units that do not exceed seven years of effort, one calendar year in length, with a system deliverable every six months or less, and no task exceeds two calendar weeks in duration. . As above, but maximum 15 years of effort, two calendar years in length. 	Add 3 points to any rating above Add 1 point		<u>Bonus Points:</u> INPUT's research indicates not only more predictable results, but fewer errors and lower total effort where these constraints are used.
<ul style="list-style-type: none"> - Estimates vary widely: <ul style="list-style-type: none"> . But causes are analyzed and used to modify future estimates. . But causes are not known or not used to modify future estimates. 	-1 -3	2-4 2-4	<u>Penalty Points:</u> Not applicable to stages 0-1, where variations will not be noticed, but a detriment to quality and to user satisfaction at later stages, if not addressed. See Bonus Points for suggested constraints.
<ul style="list-style-type: none"> • Analysis and Design. <ul style="list-style-type: none"> - The analysis and design process: <ul style="list-style-type: none"> . Uses fully integrated, computer-aided design and analysis tools, including prototypes, requirements dictionary, data dictionary, reusable code catalog, graphics and text processors, logic checkers, code 	-2 +1	0-1 2	Would be counterproductive and would probably fail. Marginally useful, not yet prepared to absorb this power.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
generators, and test harness and analyzers, for all maintenance and development projects via specialized light pen or touch panel terminals.	+10	3-4	Critically important, need to develop in-house if necessary.
<ul style="list-style-type: none"> Uses any six of the above tools, plus a recognized data flow analysis method, for all maintenance and development projects. 	-2 +1 +7	0-1 2 3-4	As above.
<ul style="list-style-type: none"> Uses any three of the above tools, plus a recognized data flow analysis method, for all maintenance and development projects. 	-1 +2 +4	0 1 2-4	Counter productive, would probably fail. Useful, no more should be attempted at this stage. Useful, can absorb more.
<ul style="list-style-type: none"> Uses a recognized data flow analysis method for all maintenance and development projects. 	+1 +2	0-2 3-4	Useful, a good place to begin. Useful, but should be aiming much higher.
<ul style="list-style-type: none"> Use of any of the above tools for maintenance and development, but without a recognized data flow analysis method. 	-½ per tool	0-4	<u>Penalty Points:</u> Tools without theory likely to be wasteful even in the short term.
<ul style="list-style-type: none"> Any of the above levels, if applicable to new development only, should have ratings divided in half. 	varies	0-4	Maintenance consumes 2/3 of programming costs.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> ● Programming Process. <ul style="list-style-type: none"> - Programming for all maintenance, repair and new development: <ul style="list-style-type: none"> . Uses only the basic structured programming constructs in a language or preprocessor that enforces and optimizes these constructs; uses highest-level language (including menu-driven language) available for task, according to defined standards of operation; uses reusable code and skeleton programs for 60% or more of all programming. . Any of the above, but not all. . Any of the above, but applying only to new development. 	<div>+5</div> <div>+1 each</div> <div>½ of above score</div>	<div>0-4</div> <div>0-4</div> <div>0-4</div>	<p>This represents a continuum that can be applied in isolation from other tools, but gains when used in combination with other structured approaches.</p>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> - Programming skills are taught via a maintenance approach, not a new development approach: <ul style="list-style-type: none"> . To all new hires, regardless of experience level. . To entry-level personnel only. . Maintenance skills are not included in the training curriculum. 	+2 +1 -2	0-4 0-4 0-4	Bonus/Penalty Points: Applicable to any stage of development; emphasize importance of maintenance effort in total software implementation process.
<ul style="list-style-type: none"> • Testing and Certification. <ul style="list-style-type: none"> - Testing and certification: <ul style="list-style-type: none"> . Of all programs (maintenance, repair and new development) is done or reviewed by a separate quality assurance group, including users; uses an automated regression test harness, stated test criteria and at least two independent testing strategies; configuration control is in place. 	-2 +8	0-1 2-4	Would be counterproductive, a waste of resources. Very important in the later stages.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Any two of the above characteristics, but still applying to maintenance, repair and new development. Any single characteristic applying to maintenance, repair and new development. 	-2 +4 +1 +2	0-1 2-4 0-1 2-4	Very important in the later stages. A lower level of activity is appropriate in these stages. Useful, but should be striving for a more complete testing process.
<ul style="list-style-type: none"> In addition to the above, statistics concerning cause, source and severity of errors are generated and analyzed. 	+2	1-4	<u>Bonus Points:</u> Would probably not be used at stage zero, but valuable above that stage.
<ul style="list-style-type: none"> Any level above, not applied to maintenance or repair. 	½ of above score	0-4	<u>Penalty Points:</u> Reflects the high cost of maintenance as compared to development.
<ul style="list-style-type: none"> Production. <ul style="list-style-type: none"> Production libraries and files: <ul style="list-style-type: none"> Are totally isolated from testing libraries and files; automated procedure is in place for publicizing and signing-off on changes. 	+5	0-4	Represents a continuum applicable to any stage.

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> As above, but manual procedure. Are not isolated, but manual procedure is in place as above. Are not isolated, and no procedure exists as above. 	+3 -1 -5	0-4 0-4 0-4	Risk of catastrophe to production files. Extreme risk of catastrophe to production files.
<ul style="list-style-type: none"> Production libraries and files are audited: <ul style="list-style-type: none"> At regular and surprise intervals. At regular intervals, at least annually. 	+3 +1	0-4 0-4	<u>Bonus Points:</u> Represents a continuum.
<ul style="list-style-type: none"> In addition, statistics concerning cause, source and severity of production failure are generated and analyzed at the detail level. 	+1	0-4	<u>Bonus Points.</u>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> ● Enhancement and Maintenance. <ul style="list-style-type: none"> - Enhancement and maintenance: <ul style="list-style-type: none"> • Are subject to the same standards of approval as new development. • Have separate, but stated and enforced, standards. • Are not covered by standards. 	<div></div> <div>+5</div> <div>0-4</div>		<div></div> <div>Represents a continuum.</div>
<ul style="list-style-type: none"> - An annual review of all existing systems is conducted to prepare a preventive maintenance plan. 	<div></div> <div>+2</div> <div>0-4</div>		<div></div> <div>Desirable at all levels; probably not achievable in 0 and I.</div>
<ul style="list-style-type: none"> - In addition, enhancement and maintenance tools are in place to provide: <ul style="list-style-type: none"> • Global cross-referencing. • Protection against reintroduction of previous errors. • Automated modification of documentation. 	<div></div> <div>+2</div> <div>0-4</div> <div>+4</div> <div>0-4</div> <div>+4</div> <div>0-4</div>		<div></div> <div>Bonus Points: Desirable in all stages; probably not achievable in 0 and I.</div>

SELF-ANALYSIS, ACTIVITY-ORIENTED, PRODUCTIVITY AUDIT	POINT SCORE	STAGE	EXPLANATION
<ul style="list-style-type: none"> Is produced after the system is operational. 	-1	0-4	Doubtful if higher stages can be attained with this approach to documentation.
<ul style="list-style-type: none"> Is dependent in quality upon the individual who prepared it. 	-2	0-4	As above.
<ul style="list-style-type: none"> Is done only when there is time. 	-4	0-4	As above.
<ul style="list-style-type: none"> Log of program errors and changes is maintained: <ul style="list-style-type: none"> Within program listing. Separately. 	+2 +1	0-4 0-4	Bonus Points: May claim any or all.
<ul style="list-style-type: none"> Users' manuals: <ul style="list-style-type: none"> Are prepared by the user as part of the system specification. Are prepared jointly by the user and the system developer. 	+2 +1	0-4 0-4	Bonus Points: May claim any or all.

APPENDIX F: QUESTIONNAIRE

GENERAL

1. How would you describe your company with regard to productivity:
 - a. Generally _____
 - b. EDP _____
- 2a. ()Is there ()Was there a productivity problem in your organization?
How would you define it?
- 2b. How does it relate to the area of software development and/or maintenance?
3. What steps have been taken to address the problem?
4. How successful have they been?
- 4a. Does your management agree with your assessment of "success"?
- 4b. Is corporate management committed to productivity improvement? Give examples.

- 4c. Do you have a management plan for where you want to be in productivity?
What and why or why not?
- 4d. Who has been specifically designated responsible for productivity improvement?
- ☐ No one.
 - ☐ Everyone.
 - ☐ Each manager.
 - ☐ An individual reporting directly to the top department manager.
 - ☐ An individual reporting elsewhere (where?_____).
 - ☐ A team composed of _____ level personnel.
 - ☐ Other. Describe. _____
- 4e. What kind of training/direction regarding productivity progress and/or measurement techniques have these individuals received?
- 5a. How would your management define software productivity improvement?
- ☐ Lower cost of development.
 - ☐ Lower total cost (including operation and maintenance).
 - ☐ Other cost-related definition. Describe. _____
 - ☐ Shorter time to implement/better adherence to schedules.
 - ☐ Other time related definition. Describe. _____
 - ☐ Higher quality product (longer MTBF, shorter MTTR).
 - ☐ Higher utility product (more functional, contributes more to ROI).
 - ☐ Higher quantity of useful output.
 - ☐ Higher ratio of output divided by input.
 - ☐ Other. Describe. _____

- 5b. How would management prefer to measure the above?
6. How would you/do you allow for differences in skill level, complexity, quality, type of application, etc.?
7. Is this satisfactory? If not, how could it be improved?
8. Do you have any statistics to support your point of view? Explain.
- 9a. How does your company measure productivity in other departments?
- 9b. Is there an EDP steering committee? If yes, describe.
- 10a. Have you experienced higher or lower productivity in software in the last few years? Explain and quantify.
- 10b. To what factors do you attribute this?

- 11a. Have you tried something and given up on it? If yes, what and why?
- 11b. Have you heard of someone who is doing something successful and/or something that failed? Who and what, and why (if you know).
12. How much of an increase in software productivity do you need?
- () Less than 5%.
 - () 5%-14%.
 - () 15%-24%.
 - () 25%-49%.
 - () 50%-99%.
 - () 100% or more.
13. What do you consider an appropriate level of output and why?
14. Relate this in specific terms to the productivity definition and measures above.
15. Where in the systems life cycle does this improvement have to come?
- () Requirements/feasibility definition ____%.
 - () Specification ____%.
 - () Design ____%.
 - () Programming ____%.
 - () Testing ____%.

- () Operation ____%.
- () Maintenance ____%.
- () Enhancement/adaptation ____%.
- () Estimation/costing techniques ____%.
- () Management control techniques ____%.
- () Quality control ____%.

Please give some examples.

16. To what factors do you attribute these needs for improvement?

Rating: 1 = very important, 5 = not important

- () Inadequate skill levels or training of personnel.
- () Inadequate skill levels or training of management.
- () Inadequate supervision/poor definition of objectives.
- () Inadequate technical resources.
- () Inadequate planning/unrealistic (optimistic) scheduling.
- () Inadequate control and feedback.
- () Inadequate personnel motivations/attitudes.
- () Insufficient definition of requirements/failure to freeze requirements.
- () Excessive complexity of requirements.
- () Failure to implement a prototype.
- () Inappropriate management style.
- () Inadequate measurements.
- () Wasted time.
- () Non-productive time.
- () Inadequate theoretical basis.
- () Conflicting priorities.
- () Trivial or non-essential work in the backlog.
- () Excessive turnover, unstable environment.
- () Other. Describe.

Please give an example to illustrate your point for each item rated 1.

17. In your opinion, what improvements are required?
18. Have any steps been taken toward this goal? What were the results?
19. Please rank the following areas for their importance in achieving greater software productivity. 1 = Highest, 12 = Lowest.
- () Environmental factors (physical facilities, organization, working hours & conditions).
 - () Personnel recruitment, selection.
 - () Personnel training.
 - () Personnel motivation.
 - () Process tools and aids.
 - () Management interest, training.
 - () Management motivation, support.
 - () Management tools (planning, estimating).
 - () Management skills (control).
 - () Corporate environment (personnel policies, etc.)
 - () Role of the end user.
 - () Other. Describe. _____

BACKLOG

20. How long is your project backlog? _____man-months _____ projects.
21. How is its length controlled?

22. Of what types and sizes of tasks is it composed?

_____%, _____#	man-months modifications or additions of reports for users.
_____%, _____#	man-months modifications or additions or reports for management.
_____%, _____#	man-months modifications or additions of functions.
_____%, _____#	man-months integration or redesign of existing systems.
_____%, _____#	man-months creation of new systems.
_____%, _____#	man-months other. Describe. _____

23. What is the total dollar value of these backlog systems to your organization, in terms of increased revenue, decreased expenses, greater market control, etc.

24. What would be their total cost to implement?

END USERS

25. What basis exists for end users to solve their own problems?
(Policy, tools, support, training, incentives, criteria for approval, etc.)

26. What drawbacks are there for allowing users to implement their own systems?
Give examples if you have already tried it.

27. How do you decide which systems, areas, etc., end users can implement directly?

28. How did you introduce and achieve this mode of operation?
29. How satisfactory are the results?
30. What changes should be made?

MEASUREMENTS

31. What productivity measurements are being used?
- ☐ None.
 - ☐ Project status reporting ☐ Daily ☐ Weekly ☐ Monthly.
 - ☐ Project cost reporting.
 - ☐ Productivity ratio reporting.
 - ☐ Other. Describe. _____
32. Who and what are included in the reporting system?
- ☐ All staff plus management hardware plus resources.
 - ☐ All staff plus resources.
 - ☐ All staff.
 - ☐ Partial staff plus partial resources.
 - ☐ Partial staff.
 - ☐ Partial resources.
 - ☐ Other. Describe _____
33. How long has the reporting system been in operation?

34. How and where are the results of the reporting system used?
35. What is the feedback mechanism?
36. What incentives for improvement are offered?
37. What specific units are measured?
- () Lines of code per day, month, etc. (Explain refinements)
 - () Number of compilations until syntax error free.
 - () Number of errors found during reviews.
 - () Types of errors found.
 - () Number of test runs.
 - () Number of turnarounds per day.
 - () Other types of counts: Describe _____
 - () Hours of error free running.
 - () Ratio of conformance to standards.
 - () Hours of development time per function delivered.
 - () Value of product as a ratio to specified value standards (ROI, etc.)
 - () Percentage of overruns in development.
 - () Cost to redo/time to repair.
 - () Other. Describe. _____
38. What related measurements are tracked?
- () Absentee days.
 - () Absentee rate.
 - () Turnover.
 - () Turnover by project, skill level, etc.
 - () Lost time as a result of resource unavailability.
 - () Non project time as a percentage of total time.
 - () Other. Describe. _____

MAINTENANCE

39. What maintenance/enhancement measurements are used?

- () Same as development.
- () Lines of code modified per day, month, etc. (Explain refinements).

- () Number of compilations.
- () Number of test runs.
- () Number of turnarounds per day.
- () Hours of maintenance/enhancement time per function delivered.
- () Ratio of conformance to standards.
- () Cost of lost time/unavailability of system.
- () Cost to redo/time to repair.
- () Other. Describe. _____

40. Do you budget and schedule maintenance time to the same set of standards as for new development?

41a. What estimating techniques do you use when planning projects?

41b. Do you consider these techniques successful and why?

42. What do you consider an optimum size for projects, and why?
43. Do you perform any software preventative maintenance (bringing systems up to current standards)?
44. Are there any techniques you apply to new development, but not to maintenance?
45. Are there any techniques you apply to maintenance, but not to new development?

PERSONNEL

46. What hours do your EDP personnel work?
- () 5 eight hour days including lunch and breaks.
 - () 5 eight hour days not including lunch.
 - () 4 ten hour days () including () not including lunch.
 - () Flextime () 35 () 37½ () 40 () _____ hour week.
 - () Other. Describe. _____
47. Is overtime:
- () Mandatory, not compensated in any way.
 - () Mandatory, compensated by time off.
 - () Mandatory, compensated in straight time dollars.
 - () Mandatory, compensated in time and a half dollars. (or more)

- () Voluntary, not compensated in any way.
- () Voluntary, compensated by time off.
- () Voluntary, compensated in straight time dollars.
- () Voluntary, compensated in time and a half dollars.(or more)
- () Not permitted.

48. Does overtime get charged against project budgets? At what rate?

- () 1 to 1 prorated at _____% of regular time () varies. Explain.

49. Is work at home () Permitted () Encouraged () Required () Prohibited?

50. How does work at home get charged to project budgets?

- () Full rate as if on-site.
- () Partial rate. Explain.
- () Not charged.

51. How is the amount and quality of work at home controlled?

52. Is EDP security an issue with work at home? If yes, how is it handled?

53. How was "work at home" introduced to your organization? Do other departments also () permit () encourage () require "work at home"?

54. What percentage of on-site work time is normally spent for project activity?
(As compared to interruptions, department meetings, etc.)

55. How is the typical work ()day ()week spent for programmer/analysts.

- _____ hours defining, consulting with user.
- _____ hours researching systems, facilities (non-people related research).
- _____ hours discussion, review with peers.
- _____ hours writing, code or specifications.
- _____ hours documenting
- _____ hours redesign, defect removal.
- _____ hours project-related work (administrative).
- _____ hours other company-related work,
- _____ hours vacation, holidays, sick, training other non-project time.

56. Do you have any statistics or data to support this breakdown? Explain.

57. Do you use this breakdown in planning/budgeting projects? Explain.

58. How many of your EDP staff work part time (less than 30 hours a week).

59a. How many of your EDP staff are hired on a consulting (body-shop) basis?

59b. Will this change in the future? How and why?

59c. Do they work under specified standards of performance? Explain.

59d. How is their productivity controlled? How does it compare to productivity of your own people?

59e. Do you use internal contract arrangements? Explain.

60. What changes in your personnel requirements and structure do you expect in each of the following years?

NUMBER REQUIRED

ORGANIZATION STRUCTURE

# _____ On board, end of 1978	_____ % centralized.
# _____ Changes in 1979	_____ % centralized.
# _____ Changes in 1980	_____ % centralized.
# _____ Changes in 1981	_____ % centralized.
# _____ Changes in 1982	_____ % centralized.
# _____ Changes in 1983	_____ % centralized.

61. What skill specialties are you looking for?

62. Where do you expect to find these people (if you are adding staff)?

_____ %	Internal transfers.
_____ %	Hired as trainees.
_____ %	Hired 1-3 years experience.
_____ %	Hired with more than 3 years experience.
_____ %	Recruited as computer science graduates (BS/BA).
_____ %	Recruited as advanced computer science graduates.
_____ %	Recruited as general business graduates.

Do you () recruit directly _____ % () use agencies _____ %.

63. How do you expect to make up for the shortfall (if any)?
- 64a. What entrance tests or other screening techniques do you use?
- 64b. How successful are they, and how do you measure their effectiveness?
- 64c. What is the salary range for your area?
- 64d. Are your salaries above, at, or below average for your area?
- 64e. What do you believe is the most single important motivation for systems people?
- 64f. What non-salary incentives do you offer?
- 64g. How do you measure the success of these incentives?
- 54h. Do you plan to use more or less incentives over the next 3 years?
How much and what kinds?

65. What is your turnover level overall? _____%/yr.
66. What is it generally in your geographic area? _____%/yr.
67. What is your turnover level within 4 years of hiring? _____%/yr.
68. By how much does this differ from other units of your organization? _____%
69. What percent turnover do you want to achieve and how do you plan to do it?
70. What effect does your present turnover level have on your productivity level?
71. To what factors do you attribute the difference between your rate and the comparable rates in your area?

TRAINING

- 72a. Do you have a formal career development plan? If yes, describe.
- 72b. How often and how do you evaluate personnel? Does this differ from the rest of the corporation?
- 73a. What training needs are not currently being met adequately?
- 73b. What are your plans to achieve adequate training?

74. How are the courses for an individual selected and approved?
75. Has your training program () increased () decreased turnover?
By how much? _____% How do you know?
76. What kind of training do you provide end users, and how much is it used?
(Use codes to indicate usage: A=more than 50% of end users, B=26%-50% of end users, C=11%-25% of end users, D=less than 10% of end users, E=not at all)
- () Self instruction manuals.
 - () A-V training.
 - () Stand up lectures.
 - () "HELP" panels built into application systems.
 - () Other. Describe. _____

77. What is your training budget for EDP staff? \$_____/year.
End users \$_____/year.

ENVIRONMENT

78. What is your operating environment?

<u>CPU</u>	<u>MEMORY</u>	<u>O/S</u>	<u>% UTILIZATION</u>
a.			
b.			
c.			
d.			

<u>% PRODUC- TION</u>	<u># TERMINALS</u>	<u># DEVEL. TERM</u>	<u>USING TSO, etc.</u>
a.			
b.			
c.			
d.			

79. What is your development terminal response time? _____ to _____ seconds.
80. How many daily turnarounds do your programmers now get?
81. How does this compare with your previous situation and when was that?
82. Do you consider your current situation satisfactory? If not, what are your plans?

ORGANIZATION

83. Draw an approximate organization chart for EDP, showing major application and systems groups, locations, and numbers of programmers, analysts, and other staff.
- 84a. How many systems analysts are assigned to user departments?
Whose budget are they on?
- 84b. How much of the budget gets offloaded to the end users now?

- 84c. How much is charged back, and on what basis?
- 84d. How will that change, and by what percent over the next 3 years?
- 85a. Are staff organized by new development versus maintenance? Are you satisfied with that arrangement?
- 85b. Do you distinguish between programmers and analysts? Are you satisfied with that arrangement?
86. What type of individuals and how many are working on maintenance?
87. What type of individuals and how many are working on new development?
88. What type of individuals and how many are working on enhancements?

PROCESS AIDS

89. Are you now using, have used, or plan to use any of the following?
If yes, please answer additional page for each.

<u>Now</u> <u>Use</u>	<u>Have</u> <u>Used</u>	<u>Plan to</u> <u>Use</u>		<u>Which One</u>
()	()	()	Formal systems design methodology	_____
()	()	()	Manual project control system	_____
()	()	()	Automated project control system	_____
()	()	()	Integrated project control system	_____
()	()	()	Requirements data base (PSL/PSA,BIAIT,etc)	_____
()	()	()	Analysis methodology (SA, DeMarco, etc.)	_____
()	()	()	Design methodology (Jackson, Yourdon, etc.)	_____
()	()	()	Design language (PDL, pseudo code, etc.)	_____
()	()	()	Flowchart alternatives (Nassi-Scheiderman,etc.)	_____
()	()	()	Automated design aid (lightpen diagrammer)	_____
()	()	()	Fault tolerant design	_____
()	()	()	Formal defect removal strategy	_____
()	()	()	Relational data base	_____
()	()	()	Mini/micro data base	_____
()	()	()	Data dictionary	_____
()	()	()	Integrated data dictionary	_____
()	()	()	Program generator	_____
()	()	()	Reusable code library	_____
()	()	()	Prototype (skeleton) programs	_____
()	()	()	Heuristic programming	_____
()	()	()	Structured programming preprocessor	_____
()	()	()	Standards enforcer	_____
()	()	()	Program optimizer	_____
()	()	()	Very high level language	_____
()	()	()	Non-procedural language	_____
()	()	()	Text editor	_____
()	()	()	Pascal, C, other structured language (not PL/I)	_____
()	()	()	Programmer work bench, Maestro, etc	_____

<u>Now</u> <u>Use</u>	<u>Have</u> <u>Used</u>	<u>Plan to</u> <u>Use</u>		<u>Which One</u>
()	()	()	Debugger, trace program, etc.	_____
()	()	()	Formal testing methodolgy	_____
()	()	()	Design reviews, structured walk thru	_____
()	()	()	Automated tester	_____
()	()	()	Instrumenter	_____
()	()	()	Regression test driver	_____
()	()	()	Other testing aid	_____
()	()	()	Maintenance aid	_____
()	()	()	Error history reporting	_____
()	()	()	Documenter (automated)	_____
()	()	()	End user development	_____
()	()	()	User friendly language	_____
()	()	()	"Quiet Rooms"	_____
()	()	()	Decentralized development (DDP)	_____
()	()	()	Matrix project organization	_____
()	()	()	Others. Specify _____	_____
()	()	()	_____	_____
()	()	()	_____	_____
()	()	()	_____	_____

Category of Aid _____	Name of Aid _____
Vendor _____	Address _____
Cost _____	Configuration _____
In use since _____	By (% , number of staff, etc.) _____

Training Required

Range of Benefits

Benefits Achieved

Time to Achieve Benefits

Ability to Measure Benefits

Justification Used

Basis for Decision

Implementation Experience

Improvements Needed

Interaction with Other Aids

Acceptability to Staff

Acceptability to Management

